UX from 30,000ft

A Guide to User Experience for Software Engineers and Developers

Simon Harper

UX from 30,000ft

A Guide to User Experience for Software Engineers and Developers

Simon Harper

This book is for sale at http://leanpub.com/UX

This version was published on 2015-04-14

Tweet This Book!

Please help Simon Harper by spreading the word about this book on Twitter!

The suggested hashtag for this book is #UX30000.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search?q=#UX30000

Contents

Preamble

Welcome to User Experience! For many of you this will be your first exposure to the wider discipline of Human Computer Interaction (HCI) and Human Factors. Some of you may have already experienced some basic HCI work as part of the software engineering lifecycle, however this text takes a far more in-depth look at the tools, techniques, and knowledge you need to understand the user experience within software engineering.

Figure: Tom's Diner. "Tom's Diner" is an a cappella pop song written in 1981 by American singer-songwriter Suzanne Vega. Vega wrote the song based on a comment by her friend Brian Rose, a photographer, who mentioned that in his work, he sometimes felt as if 'he saw his whole life through a pane of glass, and ... like he was the witness to a lot of things, but was never really involved in them' —Image Credit: Wikipedia.

It is worth noting that while this text is more in depth than those to which you may already have been exposed, it is by no means intended to teach you everything you need to know about human facing software engineering. Indeed, entire undergraduate degrees are built around the subject of human factors and ergonomics and you should not confuse this high-level overview of the domain with the knowledge you would acquire in a full three-year degree programme. This said the aim of this text is to give you, the reader, the tools, techniques, and the mindset necessary to competently approach your first user testing and user experience job. The text is designed from a practical perspective and will enable you to take a junior role in a user experience department, or usability company, and will provide you with the overall knowledge to communicate with others and make sensible suggestions regarding UX work. Further, it will provide you with a basis for future self study within the UX domain or the wider human factors world. In this case I've decided to call the text, more fully, 'User Experience from 30,000 feet'[1].

> Our need to understand the user experience manifests itself in many and varied ways; even in the most unexpected places. Go listen to the song 'Tom's Diner' by Suzanne Vega – make sure it's the a'cappella version – now do a little Web searching and answer these questions:
>
> 1. What is the significance Tom's Diner in your everyday life?

[1]UX does have one advantage in that it is a new, practical, cross-disciplinary subject. No one has yet trained on a bespoke UX only degree programme and so everyone has their own background and 'leaning' within the area. There are many UX Industrial Departments/Companies, there are few UX courses. The combination of your technical Computer Science training coupled with this UX training will give you an advantage; a software focused UX professional.

2. Why is Tom's Diner significant for the User Experience?
3. What properties of Tom's Diner makes it so significant?
4. Why does the significance of Tom's Diner represent 'Good' science?

> Now, don't panic I don't expect you to know the answers to all of these questions right now – but humour me – do some research and come up with some answers that seem plausible to you.

As we'll see later, the UX domain is still very young and in the process of formation, however, it does bring together a number of already established areas within the human computer interaction field; and as a place holder you may wish to think of UX as the practical application of research knowledge repurposed from other domains into the user facing software engineering process. In this case, suggesting a particular UX text would only serve to skew your education to a particular view of the UX domain (although you may wish to check out some UX books as well: **[Unger and Chandler, 2009**; **Bowles and Box, 2011**; **Lund, 2011]**. I feel it is better for you to understand my view, while realising there may be different ones out there and that in the end you will need to decide, after reading this text, the sort of UX you wish to do and how you think about the area. Therefore this text has been written from scratch and represents an overview of UX as seen by the human factors and web science researchers I work with.

> You don't need to panic, the text is intended to provide you with: an overview of the UX field; an understanding of the key concepts within this field; a method of assessing your understanding of the topics covered; and finally, a reference to reading material which will enable your deeper study, both now and in your future work.

Finally, I've tried to keep the jargon (and explain any I introduce[2]) to a minimum and I hope you'll see that the writing style is 'formally academic' but clear and easy to read.

Origins

> To give you an idea of how this text arose, it may be useful to give you an idea of it's origins.
>
> Similar to many Computer Science Departments, the School of Computer Science at Manchester has its roots in the Computer Group of the Electrical Engineering Department and the Computer Laboratory of the Department of Mathematics. After its formation in 1964, it continued to focus on computational hardware and low–level or embedded software. This focus resulted in the world's first stored–program computer; the Manchester Small–Scale Experimental Machine (SSEM), nicknamed 'Baby'. The School, run in the early days by such luminaries as Frederic C. Williams, Tom Kilburn, Geoff Tootill, and Alan Turing, continued to focus on the numerate, applied, and engineering aspects of Computer Science.

[2] Email me with any jargon I haven't explained, or that you think I should explain better.

> In this environment there was little focus on the human aspects of computer science, that would inevitably come to play an important role as the computer moved from the laboratory to the desktop and onward to be embedded in many more products and devices. However, some of the researchers at the School became increasingly aware that we were not addressing areas which would – we thought – become increasingly important.
>
> In light of this perception this user experience text was developed to serve as a cohesive starting point for us to be able to teach HCI and UX to Technical Computer Science, Software Engineering, Hardware Engineering, and Artificial Intelligence students. It is therefore based on the human factors research activity already established within the School over the past 15 years, in support of the embryonic teaching activities arising after 2010.
>
> Before we go any further it may also be useful to make one more point. You will notice that I use the term User Experience Specialist, usability specialist, UX'er, etc throughout the book when referring to you the reader. This is because I've found it useful to reinforce your belief that this is what you can become as you learn more about the field. I think using this terminology also makes you more likely to think of yourselves in a different way to the 'developers' that you probably are, and the engineering domain you feel comfortable within.

Text Structure

UX is an empirical discipline, whereby established principles and guidelines are applied to a software development and the correctness of that software is established via experimentation in the form of user feedback. In this case the UX specialist is mainly interested in questioning the development to ascertain its correctness.

This correctness normally exists along a number of different dimensions but briefly can be thought of as: the primary utility of the development; its effectiveness; its efficiency; and its affectiveness (Both tangibly and intangibly, represented by the concept of engagement). In this case the text structure is intended to give you the principles, tools, and techniques needed for UX work along with a critical and questioning mindset to enable you to apply these techniques effectively.

Therefore, you will learn the principles and tools needed for effective interaction with each dimension; the scientific and empirical techniques to join and apply these principles and tools to a new bespoke development; and the mindset to be critical of, and to question the outcomes of, this development to accurately establish the user experience.

Secondary 'Text'

Zen and the Art of Motorcycle Maintenance (ZAMM) **[Pirsig, 1974]** may seem like a strange text to use for a Computer Science based UX course; and it probably is. In reality, I am not interested in

you remembering anything in ZAMM – however the critical and questioning skills developed from its analysis will be of use to you. ZAMM is not really about Zen or indeed motorcycle maintenance, it's about science, quality, and rhetoric; the USA's Library of Congress describes it as:

> "Acclaimed as one of the most exciting books in the history of American letters, this modern epic became an instant bestseller upon publication in 1974, transforming a generation and continuing to inspire millions. This 25th Anniversary Quill Edition features a new introduction by the author important typographical changes and a Reader's Guide that includes discussion topics, an interview with the author, and letters and documents detailing how this extraordinary book came to be. A narration of a summer motorcycle trip undertaken by a father and his son, the book becomes a personal and philosophical odyssey into fundamental questions of how to live. The narrator's relationship with his son leads to a powerful self-reckoning the craft of motorcycle maintenance leads to an austerely beautiful process for reconciling science, religion, and humanism. Resonant with the confusions of existence, Zen and the Art of Motorcycle Maintenance is a touching and transcendent book of life."

While, Wikipedia says:

> "The book describes, in first person, a 17-day journey on his motorcycle from Minnesota to California by the author (though he is not identified in the book) and his son Chris, joined for the first nine days by close friends John and Sylvia Sutherland. The trip is punctuated by numerous philosophical discussions, referred to as Chautauquas by the author, on topics including epistemology, ethical emotivism and the philosophy of science. Many of these discussions are tied together by the story of the narrator's own past self, who is referred to in the third person as Phaedrus (after Plato's dialogue). Phaedrus, a teacher of creative and technical writing at a small college, became engrossed in the question of what defines good writing, and what in general defines good, or 'quality'. His philosophical investigations eventually drove him insane, and he was subjected to electroshock treatment which permanently changed his personality."

ZAMM is useful to us because it serves as framework for criticism while enabling us to apply UX principles, rational thought, and scientific argument to a well understood and well-respected text. As for my thoughts on ZAMM; well I might not agree with everything the book has to say, but I do think it is an important and interesting topic and discussion which needs to be had. At this point you need not worry too much about the book itself – other than the fact that you should read it. All will become obvious as the text progresses, but you may like to read a longer discussion on Zen and the Art of Motorcycle Maintenance and its usefulness for UX Teaching in the Appendix.

Now, lets get on with understanding and shaping the User Experience!

Chapter 1 - Everything is Wrong!

Or to quote 'Ted Nelson' more fully: *"most people are fools, most authority is malignant, God does not exist, and everything is wrong."*

'The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information' **[Miller, 1956]** is a very famous human factors research paper, written in the mid–Fifties and concerning Psychoacoustics. The paper is interesting because it has spawned, an often incorrectly understood, usability principle – being that our working memory can only handle seven (±) items at any one time and therefore we should only make menus or lists a maximum of seven items long.

However, paper is a summary/analysis of research findings couched in the form of bits per channel. He amusingly, relates the fact that the number seven seems to be plaguing his every move because seven seems to appear repeatedly in regard to the amount of information that can be remembered or differentiated by humans. However there are two caveats to this assertion: firstly, that the user is required to make absolute judgements of uni-dimensional stimuli and secondly that that stimuli is not clustered. This last point is quite important because using clustering means that we can remember or distinguish more than the unitary seven. For instance, we can remember seven characters in sequence, seven words in order, or seven phrases. In reality then, we have trouble differentiating uni-dimensional stimuli such as audible tones played without reference to each other, but we can differentiate more than seven tones when played in a sequence, or separately when multiple dimensions such as loudness and pitch are varied. Further, we are able to remember more then seven things within a list especially if those things are related or can be judged relatively, or occur as part of a sequence.

So this well found psychological finding (and very well written paper) that working memory can handle seven (±2) arbitrarily sized chunks of absolute uni-dimensional stimuli, becomes the often quoted but mostly incorrect usability/HCI principle that you should only include seven items in a menu, or seven items in a list… and on and on."

Indeed, this often incorrectly understood usability principle is a misconception that has become a firmly held belief. This is the 'wrongness' that Ted alludes to; and there are plenty of others, lets look at a few.

Buzz and Hype

Jacob Nielsen famously suggests that usability evaluations can be conducted with only five people, and this will catch over 80% of the usability errors present **[Nielsen, 2000]**. In reality, Nielsen added a lot of caveats and additional conditions to be met, which have been lost from the 'buzz' surrounding the five-users evaluation work. Eager usability practitioners hooked upon this five user

message as a way of justifying small studies, even when those small studies tested multiple and disjoint usability tasks. Even user studies that do not try to generalise their results need to make sure that the kinds of tasks performed are both limited and holistic. With these kinds of usability tests, Faulkner **[Faulkner, 2003]**, demonstrates that the amount of usability errors uncovered could be as little as 55%. Further, Schmettow **[Schmettow, 2012]** tells us that user numbers cannot be prescribed before knowing just what the study is to accomplish, and Robertson **[Robertson, 2012]** tells us that the interplay between Likert-Type Scales, Statistical Methods, and Effect Sizes are more complicated than we imagine when performing usability evaluations.

Shorthand

The Moon Orbits the Earth. More common misconceptions work their way into our understanding, usually because of a certain shorthand which makes the concepts more easily expressed, at the expense of introducing some minor errors. For instance the Moon does not orbit the Earth (see figure Moon's Orbit), well it kind-of does, but in actuality the Moon orbits the combined centres of gravity of both the Moon and the Earth which just so happens to be very near the centre of gravity exerted by the Earth; because the Moon's centre of gravity is so small.

In general, this misconception does not affect most people's normal everyday life, however, it is technically important – how celestial bodies interact with each other is a major area in the study of Astronomy.

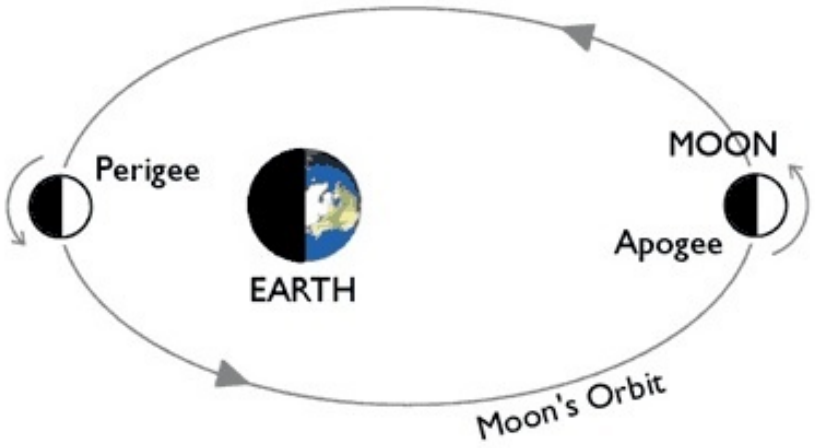

Figure: Moon's Orbit. The Moon's Orbit —Image Credit: sciencebrainwaves.com.

Sounds Right

The Earth is Closer to the Sun in the Summer than in the Winter. Let's look at another misconception, the Earth is closer to the Sun in the summer than in the winter, and it is this closeness which accounts for the seasons. Again, this is kind-of right, part of the Earth is nearer to the Sun in the summer than in the winter, however, the Earth as a whole is not. The seasons are accounted for

by the Earth being tilted on its access by 23.4˚. As the Earth orbits the Sun difference parts of the world receive different amounts of direct Sunlight and this accounts for the warming or cooling. But this warming or cooling is based upon the angle and therefore the nearness of either the southern or the northern hemisphere to the Sun, not the Earth as a whole.

In your work as a user experience specialist you often find there are many, un-said, or shorthand's expressed by users which are technically incorrect but are satisfactory to their everyday work, but may introduce large problems with your technical solutions.

Common Sense

Blind People Can't See. Certain facts seem to be 'common sense', you should always question common sense pronouncements. For instance blind people cannot see, this seems obvious, this seems like common sense, and in actuality it is incorrect. The reality of the situation is that some blind people, a very small percentage of blind people (termed profoundly blind) cannot see. In reality, the vast majority of blind people can see colours, shapes, blurred objects, and movement – in short most blind people can see something; only 4% can see nothing at all and are 'profoundly' blind.

Obviously!

Vision is Parallel, Hearing is Serial. It is often thought that vision is parallel, in that we can see multiple objects all at the same time, whereas hearing is serial because we can only hear one sound. This is incorrect, in reality the basilar membrane of the cochlear experiences different frequencies from its base to apex (see Figure: Basilar Membrane). These nerve cells are connected to different areas within the brain which are responsible to different frequencies and those frequencies are also processed in parallel.

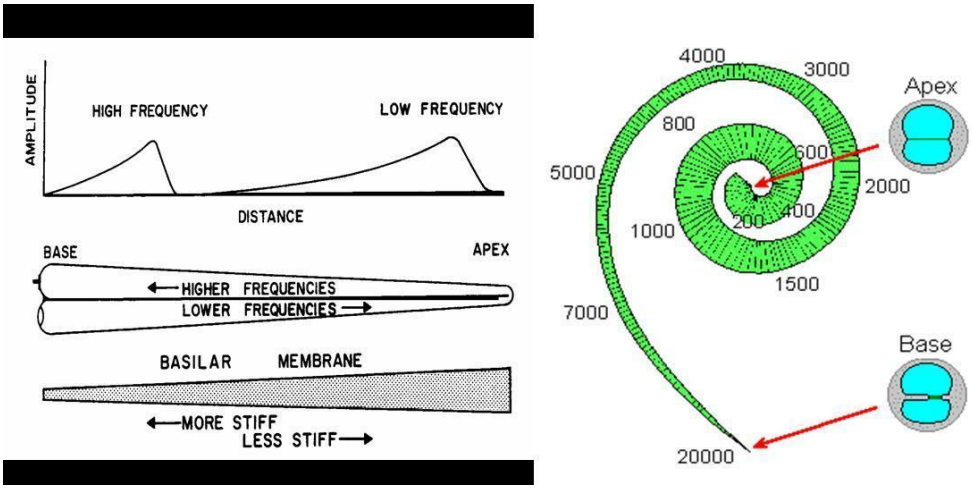

Figure: Basilar Membrane. The Basilar Membrane —Image Credit: cueflash.com.

A Little Knowledge is a Dangerous Thing

All Brains Have the Same Organisation. We all know that the brain has the same organisation, indeed, we can see this in many diagrams of the brain in which there are specific areas devoted to specific jobs, such as vision, language, hearing and the like. But just how true is this? In reality this is not as true as you may imagine, the brain develops to the age of twenty-one and within that time a concept called neuroplasticity is in effect, stating that the brain can change structurally and functionally as a result of input (or not) from the environment. This means that the brain can change and re-purpose unused areas **[Burton, 2003]**. For instance the visual cortex of a child who has become profoundly blind at, say, ten is overtime repurposed to process hearing, and possibly memory and touch. The brain develops in this way repurposing unused areas and adapting to its environment until development slows down at the age of twenty-one.

One Final Thought...

Look at this famous video from Daniel Simons and Christopher Chabris[3] which requires you to count passes of a Basketball – try to count them now, and then skip ahead for the result.

Be Curious, Be Critical

If there are two traits you should possessed as a UX specialist they are curiosity and the ability to be constructively critical. As we have seen there are many reasons why errors and misconceptions can be embedded within both an organisation, and the systems by which it runs. We can also see that if people want to believe their software has a good user experience they often will do; they are often blind to the real nature of the interface software, or systems, which they have built.

In UX there is often no 100% correct answer (that we as UX specialists can derive) when it comes to creating, building, and then testing a system. The unique nature of the human individual means that there are many subjective aspects to an evaluation and that understanding whether the interface is correct is often based on anecdotal evidence and general agreement from users, but is also based on a statistical analysis of quantitative measures. This area of uncertainty can allow rhetoric to be applied, whereby an argument may be proposed which seems stronger than your empirical evidence, but which is not empirically supportable. In addition there is also the danger that a sloppy or incorrect methodology will taint the empirical work such that the answers derived from a scientific method are incorrect. There are many cases of bad science[4], and incorrect outcomes from supposedly well conducted surveys. For instance, evidence suggests that 95 per cent of our decisions are made without rational thought. So consciously asking people how they will behave unconsciously is at best simplistic and at worst, can really mess up your study.

One of the most well known examples is the launch of New Coke. *"Coca-Cola invested in cutting-edge customer research to ensure that New Coke would be a big success. Taste tests with thousands*

[3] see http://www.youtube.com/watch?v=vJG698U2Mvo.

[4] see Ben Goldacre's http://www.badscience.net/.

of consumers clearly showed that people preferred it. The reality was somewhat different, however. Hardly anyone bought it."[5]

In another set of studies[6], people were asked what messages would be most successful at persuading home owners to make certain changes, such as turning down the heating, recycling more, and being more environmentally friendly.

> Most said that *"receiving information about their impact on the environment would make them change. However, this made very little difference at all. In another study, people who said that providing them with information about how much money they could save if they reduced consumption led to them to use even more! Interestingly, the message that most successfully changed their behaviour (information about how neighbours were making changes) was pretty much dismissed as unlikely to have any effect on them at all."*

Remember, quantitative instruments such as controlled field studies, or field observations, will often be cheaper in the long run than questionnaire based approaches. In all case, if you are not curious and you are not critical you will not produce accurate results.

HCI Foundations

> *"The human mind … operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain. It has other characteristics, of course; trails that are not frequently followed are prone to fade, items are not fully permanent, memory is transitory. Yet the speed of action, the intricacy of trails, the detail of mental pictures, is awe-inspiring beyond all else in nature.*
>
> *Man cannot hope fully to duplicate this mental process artificially, but he certainly ought to be able to learn from it. In minor ways he may even improve, for his records have relative permanency. The first idea, however, to be drawn from the analogy concerns selection. Selection by association, rather than by indexing, may yet be mechanised. One cannot hope thus to equal the speed and flexibility with which the mind follows an associative trail, but it should be possible to beat the mind decisively in regard to the permanence and clarity of the items resurrected from storage."*

This idea, first proposed by 'Vannevar Bush' in his 1945 Atlantic Monthly article 'As We May Think', is credited with being the inspiration, and precursor, for the modern World Wide Web. But for most of his article, Bush was not concerned solely with the technical aspects of his 'MEMEX' system. Instead, as with most computer visionaries, he was more concerned with how the computer system

[5] see Steve J. Martin's http://scienceofyes.com/

[6] Conducted by Wesley Schultz (Behavioural Scientist).

and its interfaces could help humanity. He wanted us to understand that instead of fitting into the way a computer interacts and presents its data, the human cognitive and interactive processes should be paramount. In short, the computer should adapt itself to accommodate human needs; not the reverse.

Since the early days of computer science, with the move from punch cards to QWERTY keyboards, from "Doug Englebart's" mouse and rudimentary hypertext systems, via work on graphical user interfaces at Xerox PARC, to the desire to share information between any computer (the World Wide Web), the human has been at the heart of the system. Human computer interaction then, has had a long history in terms of computer science, but is relatively young as a separate subject area. In some ways, its study is indivisible from that of the components which it helps to make usable, however, as we shall, key scientific principles different from most other aspects of computer science, support and underlay the area; and by implication its practical application as UX.

> We will discuss aspects of the user experience such as rapid application development and agility, people and barriers to interactivity, requirements gathering, case-studies and focus groups, stories and personas. We'll look at accessibility guidelines, and usability principles, along with emotional design and human centred design. Finally, we'll touch on the scientific method, experimentation, and inferential statistics. This seems like quite a lot of ground to cover, but it is very small in relation to the wider Human Factors / HCI domain. For instance we won't cover:
>
> - Adaptation;
> - Customisation;
> - Personalisation;
> - Transcoding;
> - Document Engineering;
> - Cognitive Science;
> - Neuroscience;
> - Systems Behaviour;
> - Interface Evolution;
> - Emergent Behaviours;
> - Application and User Agents;
> - Widget Research & Design;
> - Software Ethnography;
> - Protocols, Languages, and Formats;
> - Cognitive Ergonomics;
> - Memory, Reasoning, and Motor Response;
> - Learnability;
> - Mental Workload;
> - Decision-Making & Skilled Performance;
> - Organisational Ergonomics;
> - Socio-Technical Systems;
> - Community Ergonomics;

- Cooperative Work;
- Inferential Statistics;
- Formal Experimental Methods; and
- Mobility and Ubiquity.

User experience (UX or UE) is often conflated with usability but some would say takes its lead from the emerging discipline of experience design (XD). In reality, this means that usability is often thought of as being within the technical domain. Often being responsible for engineering aspects of the interface or interactive behaviour by building usability paradigms directly into the system. On the other hand user experience is meant to convey a wider remit which does not just primarily focus on the interface but other psychological aspects of the use behaviour. We'll talk about this in more detail later, because as the UX field evolves, this view has become somewhat out of date.

UX Emergence

Human Computer Interaction (HCI or CHI in North America) is not a simple subject to study for the Computer Scientist. HCI is an interdisciplinary subject which covers aspects of computer science, ergonomics, interface design, sociology and psychology. It is for this reason that HCI is often misunderstood by mainstream computer scientists. However, as we shall see, if HCI is to be understood and applied correctly an enormous amount of effort, mathematical knowledge, and understanding is required to both create new principles, and apply those principles in the real world. As with other human sciences[7], there are no 100% correct answers, everything is open to error because the human – and the environments they operate within – are incredibly complicated. It is difficult to isolate a single factor, and there are many extraneous hidden factors at work in any interaction scenario; in this case the luxury of a simple 'yes' or 'no' answer is not available[8].

The HCI field – of which UX is a part – is disparate, with each practitioner coming from a different specialism; in some cases psychology or ergonomics, in others sociology, for us, software engineering maybe the primary specialisation, and in the context of UX, product designers also feature. This text has its roots firmly within the mainstream computer science domain, but it is aimed at a much broader audience. Therefore, whatever your background, you should find that this text covers the principle areas and key topics that you will need to understand, and manipulate the user experience. This means that, unlike other texts on UX, I will mainly be focusing on the tools and techniques required to understand and evaluate the interface and system. While I will spend one chapter looking at practice and engineering I feel it is more important to possess the intellectual tools and skills to understand any interface you work on as opposed to memorising a comprehensive treatise of the many different interfaces you may encounter now or in the future. This is not to say that the study

[7]Defining Human science within the context of HCI is a difficult proposition especially for an empirical, small 'p' positivist, such as myself. However, I see Human science as the investigation of human life and human activities that acknowledges the validity of data derived by impartial observation within an empirical framework. It includes the subjects such as history, sociology, anthropology, psychology and economics; and while not able to validate the subjective aspect of human life and activity, is able to contribute to an understanding of the human experience.

[8]The 'up–side' is that this level of complexity makes the study of the user experience incredibly interesting and incredibly challenging if done correctly.

of past work, or best practice, is without value, but it should not be the focus of a compressed treatise such as this. In this case as technically literate readers, I expect that you will understand computational terminology and concepts such as input[9] and output[10] conventions; Graphical User Interfaces (GUIs) conventions[11]; and other general terminology; I also assume that you will know next to nothing about UX of HCI.

HCI can normally be divided into three broad stages, that of the creation of the principles theories and methodologies, the application of those aspects into a development, and the testing of the outcomes of that development. While this may sound disjoint, techniques used for the investigation and discovery of problem areas in HCI are exactly the same techniques that can be used to evaluate the positive or negative outcomes of the application of those techniques within a more production focused setting. Normally this can be seen as pre-testing a system before any changes are made, the application of those changes in software or hardware, followed by a final post testing phase which equates to an evaluation stage in which the human aspects of the interface can be scientifically derived. This pre-testing is however, often missed during 'requirements analysis', in some cases because there is no system to pre-test, and in others because there is an over-inflated value given to the implicit understanding of the system already captured.

As a UX specialist you will be concerned with the practical aspects surrounding the application of principles and guidelines into a development, and the testing of the outcomes of that development. This means that you will need to take into account the incremental nature of both the development and the experiences of the individual. Indeed we see that the user experience changes with time, and in some ways is linked with our own memory and emotional state (see 'Figure: UX Periods' and 'Figure: Time Spans of User Experience' taken from **[Roto et al., 2011]**). In this way we can see that it is possible to counteract an initially bad systems experience, in a system that possibly must be complicated (such as an aircraft cockpit), by increasing the learnability of the controls and their layout. In this case, the initial momentary and episodic user experience may be complex – and in some cases seen as negative – however, the cumulative user experience my resolve as simple – equating to a positive user experience (especially in the case of the cockpit whereby the instrumentation, systems, and their layout are often replicated between aeroplanes, regardless of manufacturer or type).

In this case how does UX relate to the standard software engineering domain of requirements analysis? The definition of requirements in the IEEE standard 610 is given in three specific points:

"(1) a condition or capacity needed by a user to solve a problem or achieve an objective; (2) a condition or capability that must be met or possessed by a system or system component to satisfy a contract standard specification or other formerly imposed document; and (3) a documented representation of a condition or capability as in (1) or (2)."

[9] Such as the keyboard, mouse, trackpad, or trackball.

[10] Such as the monitor.

[11] Such as windows, panes, panels, and the like.

We can therefore see that UX directly applies to condition '1' indeed without access to, or the usability of, a system or component by a user this condition cannot be met. In the requirements analysis domain, work often progresses in a software engineering fashion. The methodology for requirements is coalesced around a set of modelling principles often initiated by a wave of interviews, discussions, systems analysis and modelling, as well as focus group participation leading to a formal specification or model of the systems and interaction requirement[12] (requirements elicitation). These requirements have to be validated and modelled, however, as we shall see in later chapters, there are a number of problems with current approaches.

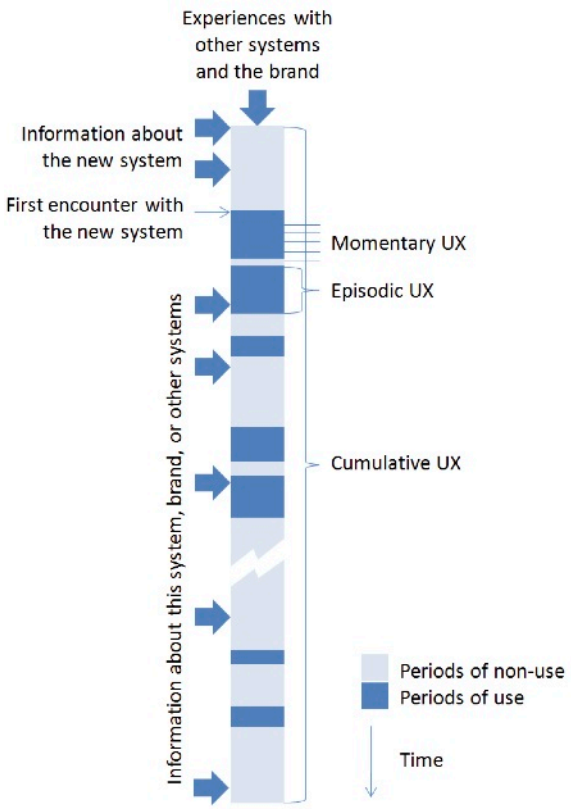

Figure: UX Periods. UX over time with periods of use and non-use —Image Credit: AllAboutUX.

The Importance of UX

While the use of HCI as a tool for knowledge discovery is important, the significance of UX in the interface design and engineering domain should not be overlooked. User facing aspects of the interface are often created by software engineers or application programmers. While these are highly trained specialists they are often less focused on aspects of user interaction than they are on the programme functionality and logic. In some cases the end user is often seen as a silent participant in the application creation process and is usually only considered once the system aspects of the development have been created and tested. Indeed, in many cases there is an implicit idea that the user will need to conform to the requirements of the system, and the interface created by the developers, as opposed to the design of the system being a collaborative activity between user and engineer.

The focus of the UX specialist then, is to make sure that the user is taken into account from the start; that participation occurs at all stages of the engineering process; that the resultant interface is fit for purpose in its usability and accessibility; and that, as much as possible, the interface is designed to fit the user and provoke a positive emotional response. By trying to understand user requirements, with

[12]Often as Universal Modelling Language (UML): Use Case Diagrams, the abstract description of user task composition and flow; Scenarios, Sequence Diagrams, and Narrative Text, the concrete description of user tasks; along with Class Diagrams, the description of user domain concepts; you probably already have experience of this from your second year software engineering course.

regard to both the system and the interface, the UX specialist contributes to the overall application design lifecycle in a very practical way which often belies the underlying scientific processes at work.

As we have seen, both requirements analysis and requirements elicitation are key factors in creating a usable system which performs the tasks required of it by the users and the commissioners of the system. Typically, however, requirements analysis and elicitation are performed by systems analysts as opposed to trained UX, human factors, or ergonomic specialists. Naturally, this means that there is often an adherence to set modelling techniques, usually adapted from the software architecture design process. This inflexibility can often be counter-productive because adaptable approaches of inquiry are often required if we are to better understand the user interaction. In reality, user experience is very similar to usability, although it is rooted within the product design community as opposed to the systems computing community of usability. Indeed, practical usability is often seen as coming from the likes of Nielson, Shneiderman, and lately Krug; while user experience came from the likes of Norman, Cooper, and Gerrett. Although, thinking also suggests these views are becoming increasingly popular[13]:

"closely coupled in practice with a deeply anti-intellectual strain that wants to remove effort, learning, and expression from computing and that valorizes the new user to the exclusion of everything else. Today's software marketplace exaggerates this: what demos well sells, and once you've made the sale, it's time to move on to the next user."

In this case, the usability specialist would often be expected to undertake a certain degree of software engineering and coding whereas the user experience specialist was often more interdisciplinary in focus. This meant that the user experience specialist might undertake design of the physical device along with a study of its economic traits but might not be able to take that design to a hardware or software resolution. Indeed, user experience has been defined as:

"pertaining to the creation of the architecture and interaction models that impact a user's perception of a device or system. The scope of the field is directed at affecting all aspects of the user's interaction with the product: how it is perceived, learned, and used."

Therefore user experience is sometimes seen as less concerned with quantifiable user performance but more the qualitative aspects of usability. In this way, UX was driven by a consideration of the 'moments of engagement', known as 'touchpoints', between people and the ideas, emotions, and the memories that these moments create. This was far more about making the user feel good about the system or the interface as opposed to purely the utility of the interactive performance. User experience then, fell to some extent outside of the technical remit of the computer science trained HCI specialist.

While once correct these views do not represent the current state of UX in the computer engineering domain. In this case, it is the objective of this text to provide you with an overview

[13] Mark Bernstein's - Creator of 'TinderBox' - view.

of modern UX along with the kinds of tools and techniques which will enable you to conduct your own well-formed scientific studies of human facing interfaces and systems within the commercial environment.

Modern UX

Defining UX is akin to walking on quicksand – there is no firm ground and you're likely to get mired in many unproductive debates – indeed, to me it seems debates on definitions are currently 'stuck in the muck'.

But why is defining UX important or even necessary? Well it must be necessary because everybody seems to be doing it, indeed 'All About UX' (AllAboutUX) have collected many definitions (see the Appendix) with multiple and different perspectives. Further, it's important because it provides a common language and understanding and a solid succinct definition enables everyone to know where they're going and – in some regard – predicts the road ahead.

So why is it such a problem? It seems to me that there is no clear definition of user experience because it is not yet a distinct domain. Everyone is an immigrant to UX[14] and there are no native UX practitioners or indeed first-generation educated practitioners who share a common understanding of what the phenomenon of UX actually is. This is always the case with new, cross disciplinary, or combinatorial domains; but this does not help us in our efforts to describe the domain such that we all understand what it is we do, where it is we are going, and what falls inside or outside that particular area.

The UX Landscape

Why am I so concerned with **Law's CHI 2009** paper and the work coalescing around AllAboutUX? The positive point about both of these sources is that they are created based on a community understanding of the area, which implicitly defines the landscape of the user experience domain. In other definitions, created by experts in the field, you are asked to subscribe to the author's interpretation. Both Law and AllAboutUX base their work on the populous view making little interpretation and allowing others to see the large differences within the comprehension, understanding, and definition of the UX field.

[14] Indeed, if you're reading this - you are probably an immigrant from the 'technical' Computer Science / Software Engineering domain.

	#6: Subjectivity		#20: Emotional attachment		#22: Qualitative approach	
	M	*SD*	*M*	*SD*	*M*	*SD*
FI	3.98	.87	2.64	1.13	3.89	.84
USA	2.93	1.09	1.93	.69	3.26	1.2
UK	3.71	1.02	2.03	.88	3.2	1.16
NL	3.47	1.16	2.00	.87	3.74	1.00

Table: UX Country Differences. Differences among countries of residence — Credit: Law, 2009.

Indeed, purely by a cursory analysis of both sources we can see that there are major differences in understanding the subjectivity, emotional attachment, and qualitative approaches, even at the coarse-grained level of countries. Indeed, 'Table: UX Country Differences' shows us that the USA sees UX as less subjective than the other countries, however all countries show a high degree of variance in the answers given; further, all countries agree on the emotional attachment aspects of user experience, but see this as being reasonably low as a factor in the UX landscape. However, all countries seem to agree that UX can be characterised by its qualitative approaches (as opposed to quantitative approaches) to understanding the experience.

#	Statement	N /275	Response Rate	M	SD	95 ci lower	95 ci upper
3	Fleeting and more stable aspects of a person's internal state (e.g.. needs. motivations) affect a person's experience of something	261	95%	4.47	.04	4.40	4.54
5	UX occurs in. and is dependent on the context in which the artefact is experienced	265	96%	4.32	.05	4.22	4.42
8	Prior exposure to an artefact shapes subsequent UX	257	93%	4.25	.05	4.16	4.34
18	Designing (for) UX must be grounded in user-centred design	265	96%	4.11	.07	3.98	4.24
23	UX can change even after a person has stopped interacting with the artefact	259	94%	3.93	.06	3.82	4.03
11	UX is based on how a person perceives the characteristics of an artefact. but not on the characteristics per se	251	91%	3.89	.07	3.75	4.03
17	UX should be assessed while interacting with an artefact	260	95%	3.87	.06	3.75	4.00
14	Measuring UX implies determination of merits, values, and significance of an artefact in relation to a person's goals and needs	249	91%	3.84	.06	3.73	3.96
13	We cannot design UX, but we can design for UX	249	91%	3.82	.07	3.68	3.96
1	UX is highly dynamic - it changes constantly while interacting with a product	264	96%	3.76	.07	3.63	3.89
12	Usability is a necessary precondition for good UX	269	98%	3.70	.07	3.56	3.84
2	Imagined use of a product can result in real experiences	235	85%	3.66	.06	3.53	3.78
15	UX refers to affective states, i.e., any combination of valence (good - bad. pleasant – unpleasant) and physiological arousal (calm – excited)	252	92%	3.60	.06	3.48	3.72
22	UX must be approached qualitatively	265	96%	3.59	.07	3.46	3.72
6	UX is not about people's performance (ability to understand and use) in their relation with an artefact, but about the person's perception of that performance	266	97%	3.58	.07	3.44	3.73
16	UX can be quantified and thus compared across similar (or competitive) artefacts	263	96%	3.50	.06	3.38	3.62
7	There is a definite need for a standardized definition of the term UX	268	97%	3.49	.07	3.34	3.63
10	UX should be assessed after interacting with an artefact	255	93%	3.33	.06	3.20	3.45
19	Only an individual person can have an experience. An experience is something personal. something 'within' a person	265	96%	3.16	.08	3.00	3.32
9	People will never have comparable UX - each and every interaction with a product results in a unique experience	268	97%	2.71	.07	2.57	2.84
21	UX is not new, it is already covered by existing engineering approaches	263	96%	2.56	.07	2.42	2.70
20	UX is equal to emotional attachment	261	95%	2.27	.06	2.15	2.39
4	UX is best viewed in terms of marketing	262	95%	1.90	.06	1.79	2.00

Table: Twenty-three Statements About UX. Twenty-three statements about UX sorted by mean agreement (M) — Credit: Law, 2009.

We can also infer (see tables taken from **[Law et al., 2009]**) that the community sees user experience

as a lens into a person's internal state which affects their experience of the software or system; that this experience must take place within the presence of the software or system they are interacting with, and that it is dependent on their prior exposure to that software or system; including it's longitudinal aspects (see 'Table: Twenty-three Statements About UX'), and that their responses are based on their 'perceptions' of the software or system they are interacting with – as opposed to the true properties of that system.

Further, the community sees that UX must be grounded in user centred design, **we cannot design UX, but that we can design for UX**. We can also see that the community do not think that UX is best viewed in terms of marketing, or that UX is only equal to emotional responses (affective), or that UX is so individual that there will never be an overlap between people and products. Indeed, extrapolating from these results we can see that most UX specialists do not see the need for a high number of users in their testing and evaluation phases, or the need for quantitative statistical analysis. Instead, they prefer a low number of users combined with qualitative output, and believe this can be extrapolated to a large population because people have comparable user experiences.

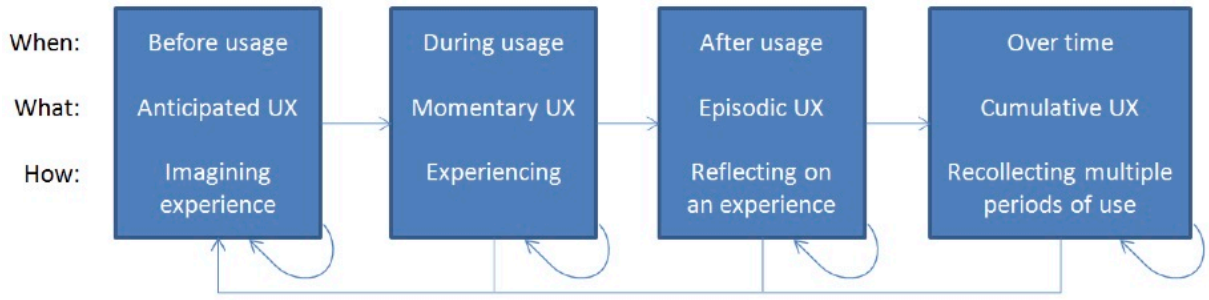

Figure: Time Spans of User Experience. Time spans of user experience, the terms to describe the kind of user experience related to the spans, and the internal process taking place in the different time spans —Image Credit: AllAboutUX.

It seems that the most important parts of the UX landscape (see 'Table: Twenty-three Statements About UX'), its nature and the ideas which are key to its understanding and application, can be summarised from the comments Law received. We could say that the nature of UX is multi-layered, and concerns the user's total experience (including their emotions and feelings) based on their current changeable internal state. UX is often socially constructed and represents the cumulative impact of interactions between users and the software or system, these interactions can be qualitatively and quantitatively measured. Key concepts through the UX landscape include the idea that not only the person, but also the artefact, and environment are equally important; and that interactions contain (un)conscious components, intangible aspects, actual and perceived interactions, and that the users entire and broad experiences are valuable.

How is the Def? (Characteristics) Positive	Negative	The Def is for? (Potential Uses)	The Def says UX is? (Nature of UX)	The Def bespeaks? (Key ideas about UX)
• *comprehensive* • *easy to understand* • *simple* • *clear* • *concise* • *accurate* • neutral • open • specific • direct • scientific • structured • system-oriented • usable • vague • descriptive • dictionary-like • high-level • integrative • memorisable	• ambiguous • circular • hard to sell • non-scientific • too academic • too broad for practice • too cognitivistic • too detailed • too dogmatic • too esoteric • too logical • too many examples • too strictly focused • wordy	• *identify all the important factors to be studied* • *enable general public to understand UX* • identify measurable aspects of UX • drive further research and development • provide a structure of UX • scoping of UX • serve as guidelines • provide a concrete set of attributes that people can relate to • provide pointers to select appropriate combination of methods for a product	• layered • lived-experience • socially constructed • task achievement • total brand experience • user's internal state • emotion • cumulative impact of interactions between users and products/services • cognitive • all feelings • experienced quality	• three dimensions: person, artefact, and environment • types of interactions: (un)conscious • value in a set of affect • intangible aspects of UX • complexity of experience • actual usage • entire user perceived experience • examples • a broad set of experiences with the company • what causes UX • not-marketing related

Figure: UX Picked Definitions. Analysis of the comments on the picked definitions —Image Credit: Law, 2009.

Caveat

As we have already seen, everything we discover should be looked at with a critical eye, including definitions and understanding of the UX landscape. So why may there be a problem with the landscape we have built in the previous section. Indeed from an analysis of Law's paper the responses from the community seemed to be reasonably strong – why should we, then, not believe them?

D1	All aspects of the end-user's interaction with the company. Its services and its products. The first requirement for an exemplary user experience is to meet the exact needs of the customer without fuss or bother. Next comes simplicity and elegance that produce products that are a joy to own, a joy to use. True user experience goes far beyond giving customers what they say they want, or providing checklist features. [8]
D2	A consequence of a user's internal state (predispositions, expectations, needs, motivation, mood, etc.) the characteristics of the designed system (e.g. complexity, purpose, usability, functionality, etc.) and the context (or the environment) within which the interaction occurs (e.g. organisational/social setting, meaningfulness of the activity, voluntariness of use, etc.) [7]
D3	The entire set of affects that is elicited by the interaction between a user and a product including the degree to which all our senses are gratified (aesthetic experience) the meanings we attach to the product (experience of meaning) and the feelings and emotions that are elicited (emotional experience). [3]
D4	The value derived from interaction(s) [or anticipated interaction(s)] with a product or service and the supporting cast in the context of use (e.g. time, location, and user disposition). [20]
D5	The quality of experience a person has when interacting with a specific design. This can range from a specific artefact such as a cup toy or website up to larger integrated experiences such as a museum or an airport. [9]

Figure: UX Five Definitions. Five definitions used in the survey —Image Credit: Law, 2009.

Firstly take a look at the demographics [Law et al., 2009]), it seems that this questionnaire was completed by far more industrial practitioners than academic ones – this may have skewed the results. Further, only 27 people said they were educated in art and design therefore we may not be getting a full view of the arts and humanities area – and how it is perceived from a non technical viewpoint. One-hundred and twenty-three people said that they were interested in understanding UX in order to design better products, suggests that product designers views, and not software technologists, may have also skewed the results. In addition look at the countries, the authors say '[people from] 25 countries [responded], with larger groups of respondents from Finland (48), USA (43), UK (36), and the Netherlands (32)' so obviously, the views from these countries will dominate. One of the main points to consider is the presumptions of the authors, as expressed in the five definitions used in the survey (see 'Figure: UX Five Definitions'). The creation of these definitions implies the authors desire to elicit broad agreement (or not) and so their creation may introduce some degree of bias, to some extent represents the authors view, and implies that this view has some priority, as opposed a choice made by the UX participants.

This said, it is still my opinion that the authors have done everything possible to be inclusive and to represent the communities view of UX, however disjoint that may be, in an accurate and informative way. By investigating the domain more fully, taking into account other sources, we too can make more accurate appraisals of the UX landscape and come to our own definition and understanding of what it means to be a UX specialist within that landscape.

My View

I've previously written about my idea of UX – having seen and read a number of definitions which suggest that UX is more about emotion and may be layered on top of other aspects of software engineering and development such as usability. However, the more I dig the more I realise that I really do not believe any of the definitions as presented, either via the excellent work of Effie Law, or those parties coalesced around AllAboutUX and led primarily by Virpi Roto.

So what do I believe?

1. I believe that UX is primarily about practice and application;
2. I believe it is an umbrella term for a multitude of specialisms;
3. I believe it is a phenomenon in that it exists and is observable;
4. I believe that this phenomenon collects people, methods, tools, and techniques from the wider human factors domain and combines them for practical application;
5. I do NOT believe that UX is a primary research domain but rather that UX is the practical application of a particular combination of tools, techniques, methods, principles, and mindset pulled in from primarily human factors and therefore psychology, social science, cognitive science, human computer interaction, and secondarily product design, and marketing;
6. I do believe that UX is a secondary field of study if the narrow definition of UX is mainly concerned with emotional indicators is used – however I believe this is more properly defined as 'affective experience'; further
7. I do not believe that UX is a 'layer' in the software artefact route to development but rather describes that software artefact in and holistic way.

Indeed, I see UX as a combination of the following properties[15]:

- **Utility**
  the software in development must be useful, profitable, or beneficial;
- **Effective in Use**
  the software must be successful in producing a desired or intended result (primarily the removal of technical barriers particularly in relation to accessibility);
- **Efficient in Use**
  the software must achieve maximum productivity with minimum wasted effort or expense (primarily the removal of barriers in relation to usability and interactivity);
- **Affective in Use**
  the software must support the emotional dimension of the experiences and feelings of the user when interacting; anticipating interaction; or when remembering interaction, with it; and

[15]...and so this is how this text is structured.

- **Engaging in Use**
  the software may exhibit an intangible dynamic deliciousness (umami) with regard to the fun a system is to use.

So, my definition (and this may evolve) would be:

> *"User Experience is an umbrella term used to describe all the factors that contribute to the quality of experience a person has when interacting with a specific software artefact, or system. It focuses on the practice of user centred: design, creation, and testing, whereby the outcomes can be qualitatively tested using small numbers of users."*

In fact on 26th January 2012 it did evolve to:

> *"User Experience is an umbrella term used to describe all the factors that contribute to the quality of experience a person has when interacting with a specific software artefact, or system. It focuses on the practice of user centred: design, creation, and testing, whereby the outcomes can be qualitatively* ***evaluated*** *using small numbers of users."*

	D1	D2	D3	D4	D5
Total	46	65	44	19	36
% out of 210	22%	31%	21%	9%	17%

Figure: Preferred Definitions. Distributions of the preferred definitions —Image Credit: Law, 2009.

Summary

As we can see, HCI is one of the most important aspects of computer science and application development, and UX is a mostly applied sub-domain of HCI; this is especially the case when that application development is focused on providing humans with access to the program functionality. But this is not the only concern of UX, indeed for many, it is the augmentation of the interactive processes and behaviours of the human in an attempt to deal with an ever more contemplated world that is the focus. This augmentation does not take the form of artificial intelligence or even cybernetics, but by enabling us to interact with computer systems more effectively, to understand the information that they are processing, and to allow us to focus more completely on the intellectual challenges; as opposed to those which are merely administrative or banal. Indeed, this objective has been Douglas C. Engelbart's overarching aim since 1962:

> *"By augmenting human intellect we mean increasing the capability of a man to approach a complex problem situation, to gain comprehension to suit his particular*

needs, and to derive solutions to problems. Increased capability in this respect is taken to mean a mixture of the following: more-rapid comprehension, better comprehension, the possibility of gaining a useful degree of comprehension in a situation that previously was too complex, speedier solutions, better solutions, and the possibility of finding solutions to problems that before seemed insoluble."

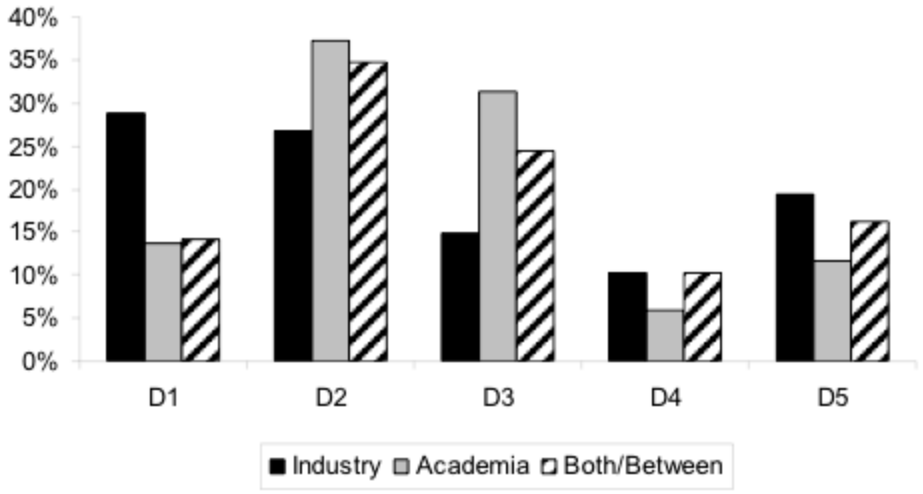

Figure: Preference by Work Place. Definition preference by the work place —Image Credit: Law, 2009.

In the context of human factors, HCI is used in areas of the computer science spectrum which may not seem as though they obviously lend themselves to interaction; indeed, even strategies in algorithms and complexity have some aspects of user dependency, as do the modelling and simulation domains within computational science.

In this regard, HCI is very much at the edge of discovery and the application of that discovery. Indeed, HCI requires a firm grasp of the key principles of science, scientific reasoning, and the philosophy of science. These philosophy, principles, and reasoning are required if a thorough understanding of the way humans interact with computers is to be achieved. In a more specific sense if you need to understand, and show, that the improvements made over user interfaces to which you have responsibility in fact real and quantifiable.

In reality, it is impossible for a text such as this to give an all-encompassing in-depth analysis of the fields which are discussed; indeed this is not its aim. You should think of this text as a route into understanding the more complex issues of user experience from a practical perspective. But you should not regard it as a substitute for the more in-depth treatise presented as part of the further reading for each chapter. In reality, the vast majority of the work covered will enable you to both develop well constructed interfaces and systems based on the principles of user experience; and run reasonably well-designed evaluations to test you developments.

Optional Further Reading

- [A. Dix] J. Finlay, G. Abowd, and R. Beale. Human-computer interaction. Prentice Hall Europe, London, 2nd ed edition, 1998.
- [C. Bowles] and J. Box. Undercover user experience: learn how to do great UX work with tiny budgets, no time, and limited support. Voices that matter. New Riders, Berkeley, CA, 2011.
- [R. Unger] and C. Chandler. A project guide to UX design: for user experience designers in the field or in the making. Voices that matter. New Riders, Berkeley, CA, 2009.
- [T. Erickson] and D. W. McDonald. HCI remixed: essays on works that have influenced the HCI community. MIT Press, Cambridge, Mass., 2008.

Self Assessment Questions

Try these without reference to the text:

- What is the key focus of HCI?
- What is the purpose of the UX specialist?
- What is User Experience and how is it applied?
- If there are no 100% correct answers in UX, how do we decide what is right and what is wrong?
- What are the five key properties of UX?

Principles of Efficient Experience

Efficient[16] use or usability[17]. To my way of thinking both terms mean the same thing, and in reality were going to be talking about usability. However, you should be thinking of usability in more general terms; the more general terms of efficient use. This is because the concept of usability is much broader than the narrow confines of 'Task Completion Time' it is often associated with **[9241-100:2011, 2011]** usability, in the context of UX, seems to be simple but in reality can be quite taxing. Complex computerised systems and interfaces are becoming increasingly widespread in everyday usage, components of computerised systems are embedded into many commercial appliances. Indeed, the miniaturisation of computerised systems, often found in mobile telephones and personal digital assistants, is on the increase.

Figure: Nest Learning Thermostat. Thermostat Showing Heating. —Image Credit: Nest Labs.

Let us take a case in point, the 'Nest Learning Thermostat' (see Figure: Nest Learning Thermostat) learns about the user and their home to balance comfort and conservation, without the hassle of programming or constant re-adjustments. Nest programs itself based on the temperatures the user sets, thereby learning the users personal schedule – in a week – and starts automatically turning down heating or cooling – to save energy – when that user is away[18]. Nest keeps refining its schedule over time, and takes input by user rotation of the outer ring to adjust the temperature; the display turns blue when cooling and red when heating, push down opens the very simple menu to enable temperature changes deletion and other fine control (see Figure: Nest Thermostat Control).

As we can see, this interface is easy to understand and operate, it uses a familiar modality – in that it it works similar to a turnable knob – and removes what has become a complicated interaction[19]

[16] Productive of effects; effective; adequately operative. The cause which makes effects to be what they are (esp. of a system or machine) achieving maximum productivity with minimum wasted effort or expense — OED.

[17] The fact or quality of being usable (That can be used; that can be readily put to practical use) — OED. The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use — ISO.}.

[18] There are other more aesthetic considerations – which we will look at later – in that the brushed stainless steel dial frames the display while the ring's curved, neutral-silver finish creates a chameleon effect that grounds Nest within its environment by picking up the colour of the wall upon which it's mounted. This combination of sleek design elements and premium materials makes Nest a thermostat a user can feel proud to display in your home.

[19] ...not least because the requirement to interact with the thermostat is sporadic; therefore reducing the familiarity which would become a learned behaviour to a bespoke interface.

(see Figure: Modern Thermostat)

into a very simple one. This occurs because the complexity of the program functionality has been removed from the end user and into the system itself. This, obviously, makes the software more difficult to develop but it removes the complexity from the user and places the onus on the software engineer and UX'er. This is important in a world where information and the requirement for its manipulation is increasing at an alarming rate, requiring the user to respond to this information and placing a premium on easy interactivity. Often however, these informational resources are centralised and therefore need to be tailored to the target audience at the time that they are delivered. Complexity, centralisation, and the need to enhance the performance of the user, means that the usability of the system, or interface, has become increasingly important.

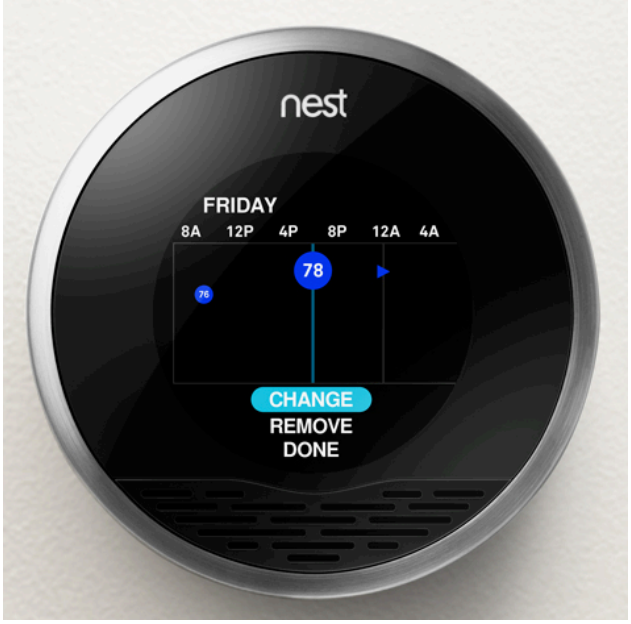

Figure: Nest Thermostat Control. Thermostat Showing Fine Heating Control. —Image Credit: Nest Labs.

Figure: Modern Thermostat. Thermostat Modern Thermostat with an 'Impenetrable' Interface. —Image Credit: Wikipedia.

The human facing aspects of the system are based on an understanding of the interactive needs of the user and the way their psychology and cognition effect their interaction. This in general manifests itself as the ability to perform tasks and actions required by the interface in ways that are also easy for the user to comprehend and execute. Indeed this is one of the primary driving forces of the graphical user interface and the mouse and keyboard. Often the usability of the system is measured by the performance, the time to execute a task, of a user enacting jobs over the system. As we shall see the rationale is, if the task is completed faster than it was before the interactive component was either altered or created when the interface design must be better as an enhancement has occurred. Common performance measures include: the time required by the user to complete a task; the time spent navigating the interface; the number of incorrect choices or errors created; the number of jobs completed, either correctly or incorrectly; the number of observations of user frustration; and finally the frequency of interface components or behaviour

that is never used.

The user centred design paradigm was created to assist in the construction of these usable interfaces. In this regard, part of the design process is driven by users who are conscripted on to the design team. In this way, it is hoped that user-friendly systems can be built and the usability of systems increased. Unfortunately, it is often difficult to solicit a representative cross-section of the possible user group, and so usability for all is often missed. This has led to the popularity of universal usability especially within the information society. In this case, universal usability refers to the design of information and communications products and services that are usable for every citizen; discussed to some extent and expanded upon later.

In reality, there is little consensus regarding the relationship of UX, ergonomics, or human factors to usability. Indeed, some think of usability

> "… as the software specialisation of the larger topic of ergonomics. Others view these topics as tangential, with ergonomics focusing on physiological matters (e.g., turning a door handle) and usability focusing on psychological matters (e.g., recognising that a door can be opened by turning its handle)."

However, a number of experts have written separate, but overlapping, frameworks for aspects of usability which should be taken into account when designing and building systems interfaces (we'll discuss these further). However, before we start on these comparisons let's look at work which gave rise to some of the earliest principles by which we should design interfaces.

The Xerox 'Star'

The Xerox 'Star' was a commercial version of the prototypical Xerox Alto – if one thousand fully working systems, used internally at 'PARC' day-in-day-out over seven years, can be said to be prototypical[20]. While the system itself is an interesting development in computer science, the interface and the mode of user interaction is truly visionary. Star, adopted novel technologies such as the mouse (the Alto had both a mouse and a portrait monitor), operationalised developments such as the graphical user interface, and created novel technologies such as the desktop (see Figure: Xerox Star Desktop). However, as UX'ers we are more interested in the design methodology they used and the principles derived from the teams experiences designing and developing the interface and interactivity of the implementations. These are some of the first usability principles to be mentioned in common computer literature – dating back to the early 1970s – and are still applicable today. First of all let us look at the design methodology adopted by the Xerox Star team – it needs no further discussion from me and so I reproduce it verbatim:

> "One of the most troublesome and least understood aspects of interactive systems is the user interface. In the design of user interfaces, we are concerned with several issues: the

[20]PARC definitely 'Eat Their Own Dog Food'!

provision of languages by which users can express their commands to the computer; the design of display representations that show the state of the system to the user; and other more abstract issues that affect the user's understanding of the system's behaviour. Many of these issues are highly subjective and are therefore often addressed in an ad hoc fashion. We believe, however, that more rigorous approaches to user interface design can be developed…"

"These design methodologies are all unsatisfactory for the same basic reason: they all omit an essential step that must precede the design of any successful user interface, namely task analysis. By this we mean the analysis of the task performed by the user, or users, prior to introducing the proposed computer system. Task analysis involves establishing who the users are, what their goals are in performing the task, what information they use in performing it, what information they generate, and what methods they employ. The descriptions of input and output information should include an analysis of the various objects, or individual types of information entity, employed by the user…"

"The purpose of task analysis is to simplify the remaining stages in user interface design. The current task description, with its breakdown of the information objects and methods presently employed, offers a starting point for the definition of a corresponding set of objects and methods to be provided by the computer system. The idea behind this phase of design is to build up a new task environment for the user, in which he can work to accomplish the same goals as before, surrounded now by a different set of objects, and employing new methods."*

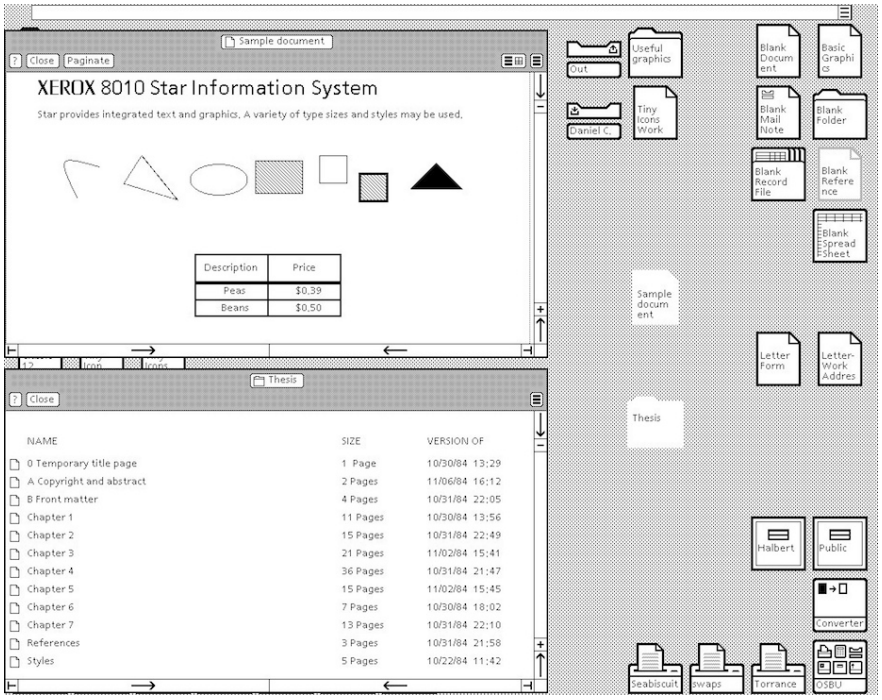

Figure: Xerox Star Desktop. Xerox Star Desktop; for a system designed 40 years ago, how different is this from our modern desktops? —Image Credit: http://toastytech.com.

As the prototyping proceeded the team evolved a number of principles: familiar user's conceptual model; seeing and pointing versus remembering and typing; what you see is what you get; universal commands; consistency; simplicity; modeless interaction; and user tailor-ability. We have already seen that the Nest Thermostat exhibits 'familiar user's conceptual model' in the use of the turning and pressing knob, both 'consistency' and 'simplicity' of operation in the movement of complexity from the user-domain, and 'tailor-ability' in the learning algorithms of the underlaying programme logic. We'll further expand upon the core Star principles next with the exception of 'seeing and pointing…', 'what you see is…', and 'modeless interaction' as these three can be taken for granted my the modern UX'er – their integration in to the operating system being complete and 'invisible':

1. 'Familiar Conceptual Models' – Star equates familiar conceptual models specifically with the idea of the desktop and the items that you might find on it. It exemplifies e-mail, and suggests that because e-mail is similar to postal mail then icons which describe e-mail in the form of in-baskets and out-baskets – letter icons being added to each to represent the incoming and outgoing mail. In addition, Star tailors these to the users familiar conceptual models by sending both text and files. This was a very new departure, previously files were sent using FTP – or some other transfer protocol – but this separation is not the case in the real world. So in reality, Star takes familiar real world concepts and maps them to the virtual world.
2. 'Universal Commands' – Star also proposes the idea of universal commands. Previously each application defined its own set of commands, and shortcuts to those commands, and expected the user to remember the specific operations without reference to those already present on the system. Star does away with all this by defining a set of universal commands

such as ‘`move`’, ‘`copy`’, ‘`delete`’ and the like; having the same action and being controlled by the same shortcuts, as every other program within the operating system. This may seem obvious and normal to us now, but it wasn’t in 1980. Now, universal commands are present in the standard operating system but there is sometimes tendency to change these commands in the application, or define a new set of commands when an application is going to be written once had distributed to different platforms (different platforms having different universal commands). In this case, you should think very hard about the rationale of breaking universality; if you are writing for many platforms, think about defining a set of commands to be used at runtime based on the kind of platform the application is being run upon.

3. ‘Consistency’ – Consistency, or pragmatic consistency as the Star Team suggest is best, is another important principle for usability. By making sure that consistent actions occur across the programme functionality we can assist the user in learning the correct outcomes for the same action over different applications, reinforce the users understanding of what these outcomes will be, match a user’s expectations with real-world actions, and dovetail into the familiarity of actions in the user’s conceptual model. Consistency should also be considered specifically with regard to the dominance of real-world physical metaphors when compared to virtual ones, and the pragmatics of maintaining consistency even when that might go against the users understanding of what should happen. In should, you need to think very carefully about how to maintain consistency within your applications.
4. ‘Simplicity’ – Simplicity is incredibly important, but it’s not a clear-cut principle; we can make a simple interface, but it may not fill the desired outcomes required by the user. What is simple to one user might be verbose and tedious to another user, especially when the first user might be a novice while the second, an expert. We can overcome some of these problems of simplicity by enabling expert access to system complexity, similar to the Nest Thermostat; we can choose to use the technique of progressive disclosure; or we can separate out novice and expert interfaces such that shortcuts and accelerated behaviours are accessible by experts – if they know how – while slower more progressively disclosed interactions which support learnability and familiarity are available for the novice user.
5. ‘User Tailor-ability’ – Usability is all about customisation, personalisation, and system adaptation; this means flexibility. We have briefly discussed the principle of flexibility already. The Star system is reasonably straightforward in its customisability, and individualisation (as it is known in the ISO standard), but this level of adaptability was unheard of at the time and pointed the way to a deeper acknowledgement that all individuals are different.

It is my opinion that these five principles are so important and timeless that their formulation and practical application as part of the Xerox Star user interface was without doubt revolutionary. Without this interface there would be no Apple Mac, or Microsoft Windows – well at least not as we know them. However not everything is down to the Star team, some of the knowledge (particularly regarding human cognition and behaviour) even predates its development, and now exist as user models to be applied before development; in an attempt to uncover the usability of an as yet unknown system.

Universal Design and Design for All!

> *"The Star system is reasonably straightforward in its customisability, and individualisation (as it is known in the ISO standard), but this level of adaptability was unheard of at the time and pointed the way to a deeper acknowledgement that all individuals are different."*

This is what Star implies in its discussion of 'User Tailor-ability'... '*individuals are different*'. The concept of universal design, or design for all, was created to address this very ideas of individual differences; and in reality means universal usability, the design aspect being applied to signify that this universality must be thought of from the design stage through to inception. Universal design can mean many things to many people. Some discuss it in terms of the society at large, by making reference to socio-economics, ethics, and issues of general discrimination. Others see design-for-all as a technological issue and a problem to be solved. Still others link design-for-all to a way of thought that should encompass everyone. In the context of computing and software development, many suggest that technology must focus on designing products so that they are usable by the widest range of people. Yet, in reality, every person is a unique individual and so this view may not be sustainable or achievable because, to create universal usability by designing for all, involves making generalisations about users, and it is these exact generalisations which were the impetus for Universality in the first place.

However, 'Universality' suggests to most UXers and engineers that the solutions they come up with must best fit most of the population most of the time. Many organisations follow the viewpoint that universal usability means design-for-all; the argument often following thus:

> *"A focus on designing products so that they are usable by the widest range of people operating in the widest range of situations as is commercially practical". Paradoxically, they also come up with a pointer to a workable solution: "As one might imagine, there are no universally usable products. There simply is too great a range of human abilities and too great a range of situations or limitations that an individual may find themselves in".*
>
> *But unfortunately do not take it: "Thus, universal usability is more a function of keeping all of the people and all of the situations in mind and trying to create a product which is as flexible as commercially practical, so that it can accommodate the different users and situations."* **[Vanderheiden, 2000]**

While Universal Usability seems reasonable on first inspection, it may not be supportable in practice. By trying to address all user needs in one design the technologist is apt to address none. Making software usable is not just about a utilitarian view of software use, it is also about the personal choice of the user (we have already touched on this). Too often designs are implemented based on knowledge and aesthetics of the designers and engineers, but not on those held by the user.

Consider the example of a spreadsheet application, in which a visually impaired user may have difficulty in choosing cells and accessing spreadsheet functions because the user interface (in this case a GUI display) does not support their needs. Another example concerns how to notify a deaf person working in a communal office that the application they are using has sound (which may be set too high).

When mainstream computer systems were initially developed, the overriding aim was focused on the creation of a computing resource, and not on the interaction of humans with that resource. Engineers and Systems Architects thought that they had overcome so-many problems in the creation of computing resources that users would be thankful just for the computational ability; *and they were*. However, as systems became more complex and machines spread to the desktop to be operated by non-technical users, interaction with these machines became more of a concern. Some solutions, collectively known as User Interface Management Systems, were suggested. These mainly focused on interface generation for large bespoke embedded systems (aircraft avionics and the like) which met with little success when moved to off-the-peg software development.

I think that system flexibility is the only way to overcome the constraints placed on users by the need to service the perceived interaction requirements of all users. Design-for-all is only needed if that design is trying to fulfil all the gaps in technology provision created by an inappropriate user interface. We need a way to make the user interface bespoke to the individual user and I think that universal access to software applications does not truly exist because the user interface and the system application logic are con- joined. Further, a stable and usable interface specification between these parts does not exist and this means that separation cannot occur. Interaction design 'Heuristics that support a separation between the user interface and the code that implements the functionality of the application are ably demonstrated in the Mozilla User Interface (XUL) implementation, which allows a different 'Look and Feel' to be used over different applications and operating systems (also see 'Skins'). By this separation, universal access to applications can be accommodated because the interface can adapt to the task without the need to change any part of the other functionality. We can see that this kind of design thinking supports the utilitarian user requirement, user activity, and users personal choice. It provides a halfway-house 'coarse' grained interface and a method of fine tuning to a specific user. Even though the rendering is still visual, Mozilla points the way to component separation, and therefore interface adaptability.

In his article 'Bridging the Digital Divide with Universal Usability' **[Shneiderman, 2001]**, Ben Shneiderman asks the question 'Can you design a text-only interface that conveys the contents and experience of an animated Flash presentation?' It is an interesting problem, but one that cannot be solved by designing one all encompassing solution. Indeed, there can only be a solution if the information in question provides the opportunity for universal access. Once the opportunity is provided, interfaces can be developed to access that information in the most appropriate way to the user and not to the information itself. For instance, if an audio file is created without the opportunity of both a graphical or textual expression, then a driver interacting by audio only could not access that information source.

My point is, you should think of Universal Design, Design for All, or Universal Usability less in terms of creating a physical design to be captured in programme logic, but more about providing

the opportunity for universality. The way to provide this opportunity, is to support 'openness' and 'consistency' of the data and API, and taylor-ability of the interface. Now it would be wrong to suggest there is no reason to provide an interface beyond an open API, we have to interface with the system at some point, but adding this separation of interface and programme logic will help you focus on supporting the users experience. Further, understanding the user, and the usability of your system will also help you build a flexible, but coarse grained interface, while understanding which aspects most require the ability to be personalised.

Usability Models

Models of the user have existed for a number of years. Some of the first where developed by Miller in an attempt to apply information theory to the human. Information theory is a branch of applied mathematics and electrical engineering involving the quantification of information. Historically, information theory was developed by Shannon to find fundamental limits on signal processing operations such as compressing data and on reliably storing and communicating data. Miller extended this model into the psychology domain by defining a model of the bandwidth that people could cope with, calling it a channel. This lead to the historic, and still quoted, 1955 work 'The magical Number Seven, Plus or Minus Two - Some Limits on Our Capacity for Processing Information' – as we have already seen). Extending this idea of modelling the user, Card et al's famous 1983 work 'The Psychology of Human-Computer Interaction' **[Card et al., 1983]** proposed the Human Processor Model which is also still referred to today. User models are in continued development with a number of active researchers trying to add knowledge and principles to the field.

You're likely to see these models variously described as User Models, Cognitive Models, or User Simulators; I've listed them here as Usability Models, because in reality that is just what they are. They see the users as a very one dimensional entity – analogising them to a processor or processing entities. These models say nothing of the non-task related traits such as effective use, emotion, or pleasure – but see the best outcome only as measurable and efficient. Keep this in mind when you are applying them, because they are my no means complete, or rich enough to do a human full justice. However they can be very useful for predicting usability.

In reality you're unlikely to come across these kinds of user models in your UX work, they are used more widely in research and at the very edges of development. This said, you may come across them in some unlikely cases, and it may also be useful for you to have a brief understanding of the key points of their application, just in case…

The Human Processor Model is a cognitive modelling method used to calculate how long it takes to perform a certain task. This method uses experimental times to calculate cognitive and motor processing time. The value of the human processor model is that it allows a system designer to predict the performance with respect to the time it takes a person to complete a task without performing experiments directly. In reality, this means that empirical work already undertaken is captured as a set of principles which are so general that they can be applied to any usability problem within their scope and domain. This removes the onus on software engineers and UX specialists to run

their own experiments, instead asserting conformance by applying the user model to their own development. The model uses the cognitive, perceptual, and motor processors along with the visual image, working memory, and long term memory stores. Each processor has a cycle time and each memory has a decay time. Therefore by following input pathways and combining them with the associated cycle or decay times, the time it takes a user to perform a certain task can be calculated. Card et al define the method for determining processes as the following steps: (1) write out main steps based on: a working prototype, simulation, step by step walk-through of all steps; (2) clearly identify the specific task and method to accomplish that task; (3) for each final step identify sub-levels down to a basic process; (4) convert into pseudo code; (5) list all assumptions; (6) determine time of each operation; (7) determine if operation times should be adjusted; (8) sum up execution times; and finally (9) iterate as needed and check with prototyping.

Goals, Operators, Methods, and Selection rules (GOMS) was also proposed in the same book as the Human Processor Model. It reduces a user's interaction with a computer to their primary actions. Using these primary actions as a framework an interface can be studied. There are several different GOMS variations which allow for different aspects of an interface to be studied and predicted. Goals are what the user intends to accomplish. Operators are actions that are performed to get to the goal. And methods are sequences of operators that accomplish a goal. There can be more than one method available to accomplish a single goal, if this is the case then selection rules are used to describe when a user would select a certain method over the others. The GOMS method is not necessarily the most accurate of all the usability measurement methods but it does have certain advantages. An estimate of a particular interaction can be calculated with minimal effort in a short amount of time. With a careful investigation into all of the detailed steps necessary for a user to successfully interact with an interface, the time measurement of how long it will take a user to interact with that interface is a simple calculation. The main drawback of GOMS is that it expects users to follow logical routines and is not resilient to user unpredictability, and all of the techniques work under the assumption that a user will know what to do at any given point. These aspect of user interaction are commonplace and are often what makes UX so challenging and interesting.

Keystroke Level Modelling, sometimes referred to as KLM or KLM-GOMS, is an eleven step method that can be used by software engineers seeking ways to estimate the time it takes to complete simple data input tasks using a computer and mouse. By using KLM, individuals often find more efficient or better ways to complete a task simply by analysing the steps required in the process and rearranging or eliminating unneeded steps thus, (1) obtain a working prototype of computer interface or a step by step operational description of a task; (2) identify the goals or the desired outcome of work; (3) for each of these goals, find subgoals or tasks that achieve the main goals; (4) identify methods to main goals and all subgoals; (5) convert description of methods to pseudo-code; (6) state any and all assumptions used in the making of pseudo-code and goals; (7) determine appropriate mental or keystroke operators for each step; (8) assign time values to mental or keystroke operators; (9) add up execution times for operators; (10) adjust total time of task to be sensitive by age of expected, this step was initially implied but is explicit as a later extension; and finally, (11) verify the validity of the results. the method id designed to be much easier than GOMS and especially useful in the evaluation of time specific tasks that require, on average, less than 5 minutes to complete.

Collated Usability Principles, Guidelines, and Rules

The first thing to notice about usability principles is that there are many of them and many different ones are proposed by many different usability luminaries[21]. This is different in some regard to those previously proposed where there is broad agreement between experts, and a smaller set of principles at work. However, usability has been a major topic in the understanding of user experience for a much longer period than has accessibility. In this case it is appropriate to collate the different usability principles along with their proponents in an effort to identify the overlap which exist between them.

You will notice in 'Table: Collated Usability Principles' that the left column describes the principle, guideline, or rule (these are sometimes used interchangeably between the different experts)[22]; while on the right side the experts are listed along with a footnote pointing to the text from which the principle is derived. In collating these principles I have not followed slavishly the nomenclature proposed by each author, but have instead placed them in categories which I believe have the same conceptual value even if the naming of that principle does not follow that in the source it is derived from.

Table: Collated Usability Principles. Usability Principles Collated by Source.

Principle	Appears in Source
Closure (Dialog Yields)	Shneiderman[23]
Consistency / Standards	Xerox-Star[24]; Shneiderman; Norman[25]; Nielsen[26]; ISO 9241-110[27]; Dix, Finally, Abowd & Beale[28]; Raskin[29].
Constraints (Exploit)	Norman.
Control & Freedom (Support)	Shneiderman; ISO 9241-110; Nielsen.
Error Handling (Simple)	Shneiderman; Norman; ISO 9241-110; Nielsen.
Familiarity & Metaphor	Xerox-Star; Norman; Dix, Finally, Abowd & Beale; Raskin.
Feedback (Informative)	Shneiderman.
Help & Documentation	Nielsen.

[21] Remember, usability principles sell books!

[22] We'll call them Principles from now on.

[23] B. Shneiderman and C. Plaisant. Designing the user interface: strategies for effective human-computer interaction. Addison-Wesley, Boston, 5th ed edition, 2010.

[24] D. C. Smith, C. Irby, R. Kimball, B. Verplank, and E. Harslem. Designing the star user interface. BYTE, 7(4):242–282, 1982.

[25] D. A. Norman. The design of everyday things. Basic Books, New York, 1st basic paperback edition, 1988.

[26] J. Nielsen. Usability engineering. Academic Press, Boston, 1993. Nielsen also lists Aesthetic and minimalist design.

[27] ISO/TR 9241-110:2006. Ergonomics of human-system interaction – part 110: Dialogue principles. TC/SC: TC 159/SC 4 ICS 13.180, International Organisation for Standardisation (ISO), Geneva, Switzerland, 2006.

[28] A. Dix, J. Finlay, G. Abowd, and R. Beale. Human-computer interaction. Prentice Hall Europe, London, 2nd ed edition, 1998. Dix, Finally, Abowd & Beale describe '*Learnability*' as: Predictability, Synthesizablity, Familiarity, Generalizabilty, and Consistency. '*Flexibility*' as: Dialog initiative, Multi-threading, Task migratabilty, Subsitutivity, and Customizability. '*Robustness*' as: Observability, Recoverability, Responsiveness, and Task conformance.

[29] J. Raskin. The humane interface: new directions for designing interactive systems. Addison Wesley, Reading, Mass., 2000.

Principle	Appears in Source
Interrupts (Resumption)	Raskin.
Describing (Self)	ISO 9241-110.
Heuristic Evaluation	Nielsen.
Learnability	Dix, Finally, Abowd & Beale, ISO 9241-110; Sharp, Rogers and Preece[30].
Mappings (Real-Virtual)	Norman; Nielsen; Dix, Finally, Abowd & Beale.
Memory Load (Reduce)	Shneiderman; Sharp, Rogers and Preece.
Navigation & Freedom (Support)	Raskin.
Reversal of Actions (Easy)	Shneiderman; Nielsen; Dix, Finally, Abowd & Beale.
Safety	Sharp, Rogers and Preece.
Shortcuts (Provide)	Shneiderman.
Simplicity	Xerox-Star; Norman; Brooks[31].
Singularity of Focus (Attention)	Raskin.
Task Suitability & Conformance	Dix, Finally, Abowd & Beale; ISO 9241-110.
Tailor-ability / Flexibility	Xerox-Star; Nielsen; ISO 9241-110; Dix, Finally, Abowd & Beale.
Universal Commands	Xerox-Star; Dix, Finally, Abowd & Beale; Raskin.
Utility	Sharp, Rogers and Preece.
Visibility (Make Things)	Norman; Nielsen.

From a cursory view of 'Table: Collated Usability Principles' we can see that there are a number of principles which deserve greater investigation including: learnability, is it easy to learn, or work out, how the system operates; efficiency, how fast can tasks be accomplished; memorability (memory load), is it easy to remember have system operates; Errors, how many errors are made and can may easily be recovered from; and satisfaction, do users like the overall feel the system. In addition I would add the following: information flow; is feedback fast, appropriate and not too detailed, but detailed enough; granularity, is the interface sectioned enough so that progressive disclosure of the interface can occur; and egocentricity, is there user specific feedback and guidance (but we will get to these later). For now however consider 'Table: Collated Usability Principles' in more detail, and I will try to explain my rationale for the grouping and discarding of the principles within it, to form a smaller aggregate set.

So I'm going to discard some of these principles because I think they are very niche, because I don't agree with them, because I don't think they are helpful, or because there seems to be little consensus in the community.*

1. First, I'm going to discard Shneiderman's 'Closure (Dialog Yields)', I think this is pretty obvious and is now a well understood software engineering practice; you wouldn't normally

[30]H. Sharp, Y. Rogers, and J. Preece. Interaction design: beyond human-computer interaction. Wiley, Chichester, 2nd ed edition, 2007. Preece, Rogers and Sharp also lists Satisfying, Enjoyable, Fun, Entertaining, Helpful, Motivating, Aesthetic, Supports creativity, Rewarding, and Emotionally fulfilling.

[31]F. P. Brooks. The mythical man-month: essays on software engineering. Addison-Wesley Pub. Co., Reading, Mass., anniversary ed edition, 1995.

have a dialog which opens another, further in some cases such as a wizard this principle isn't helpful.

2. Next I'll discard Nielsen's 'Heuristic Evaluation', not because there is anything wrong with it, but because I'm going to cover this in more detail later.
3. Next, Preece, Rogers and Sharp's 'Safety' is for the chop, while safety at the interface level maybe useful, I think it is a little too niche to be included in a general set of UX principles.
4. Now I think 'Task Suitability & Conformance' – proposed by Dix, Finally, Abowd & Beale, and ISO 9241-110 – can actually be referred to earlier, it has already been addressed as part of the UCD aspect of the design.
5. Finally, I'm discarding Preece, Rogers and Sharp's 'Utility'. Utility is obviously important - but if you've gone to the trouble of starting the development I'd imagine you expect the software to have some utility, and if the utility is about the interface under construction, well that can only be assessed after evaluation.

Now the 'carnage' is over, let's look at the principles which will stay and/or be amalgamated:

1. 'Consistency / Standards' + 'Universal Commands'. This seems pretty obvious to me - in that universality is directly related to standards to provide *consistency*.
2. 'Constraints (Exploit)' + 'Memory Load (Reduce)' + 'Simplicity'. Again I think this is straight forward, in that the constraint reduces complexity, which reduces the load on short term memory and assists in interface *simplicity*.
3. 'Control & Freedom (Support)' + 'Shortcuts (Provide)' + 'Tailor-ability / Flexibility'. Supporting user control, and user freedom, all contribute towards *Flexibility* (Tailor-ability), providing shortcuts is one additional way of providing the control we envisage. Now I'm not expanding on this one further in this chapter but instead I'm going to refer back.
4. 'Error Handling (Simple)' + 'Feedback (Informative)' + 'Singularity of Focus (Attention)' + 'Visibility (Make Things)' + 'Navigation & Freedom (Support)'. Now we don't talk about it here, but these four principles can all be subsumed to assist in '*Situational Awareness*'. Situational awareness involves a perception of the environment critical to making decisions in complex and dynamic situations. Indeed, situation awareness *"involves being aware of what is happening in the vicinity to understand how information, events, and one's own actions will impact goals and objectives, both immediately and in the near future"*. I contend that simple error's, informative feedback, visible components and interactions, coupled with a singularity of focus, all go towards creating an environment which enhances situational awareness.
5. 'Familiarity & Metaphor' + 'Mappings (Real-Virtual)'. By ensuring there is a cognitive mapping from real to virtual we can assist in the side goal of interactive *familiarity.
6. 'Help & Documentation' + 'Describing (Self)'. It is my opinion that *self description* is the principle to which good help and good documentation contribute.
7. 'Interrupts (Resumption)' + 'Reversal of Actions (Easy)' . Reversal and resumption of actions, and interactions seem to be related to me. This also feeds into user control, but these are a little more specific and seem to me to me mainly about '*interaction stability*'.

8. '*Learnability*'; we'll get to this later on.

Before moving on to a discussion of our principles, consider for a moment the usability requirements of the novice and expert user. The usability principles as described in this section not straightforwardly applicable because there is some conflict between the requirements of different users; at a course grained level these can be classed as the novice and the expert. The thing to remember with all these principles is that you need to make provision for slow, simplistic and constrained novice use; provision for fast, complex and unconstrained expert use; and some learnability path which enables novices to become experts in the shortest amount of time possible. This last requirement is most difficult because understanding if learnability is effective and efficient can only be done with longitudinal studies; in the real world it is very unlikely you will have the budget for this. This said, it is up to you as the UX usability specialist to keep these concepts of novice, expert, and the transition between the two, in your mind at all times.

As is the case for all user experience it can be difficult to validate the effectiveness and efficiency of the usability principles you are design into your system. It is often easier to see that a usability principle is not present that to see if a usability principle is present.

Potted Principles of Efficient User Experience

Interaction design focuses on improving technologies that provide interaction with the user **[9241-110:2006, 2006]**. This is normally led by the UX interaction developers, who are responsible for engineering software that shapes the user interface. Technologies mainly include Windowing Toolkits, Graphical User Interfaces, and System operations etc. Use of these technologies affect how people interact with the system both as creators and consumers of information. Therefore in any effort to support the user, it is crucial that the features and limitations of these technologies are clearly understood. For the UX'er, current core principles that should run through all aspects of the development can be summarised as follows.

Facilitate Consistency

Consistency is a key factor in usability; and two major ways of attaining consistency is to follow standards, and to develop systems which apply the same command, control, and interface structures universally across the development. While standards and universality are two key elements of consistency, the concept of consistency can exist upon its own, and you should also think at all stages of development 'Am I developing a consistent interface, are the interactions consistent across the platform and development?'

The use of standardised components is a major aspect when designing for the user. In this way the developer can be assured that look and feel of the software will be harmonious with the best practice of the operating system and dovetail into the guidelines of the underlying environment. This is often difficult if the design is to be accomplished on different platforms because the look and feel changes

(for instance there is a significant difference between the operating modality and look and feel of Microsoft Windows and Apple Mac OS X). However there are some ways to address the problems of standardisation across the interface. The developer can design for the interpreter platform, or by choosing a cross-platform systems such as Java or Qt which has adapts to the base operating system environment. In this way, the development cost is removed from the developer to the language or framework engine. In addition, the developer may wish to use techniques such as Model-View-Controller (MVC) to separate the presentation from the programme logic and use a window toolkit to enable a more harmonious distribution across different platforms. In either case standardisation to the underlying environment is critical to maintain the usability of the development and to reduce the additional cognitive overload on the user in switching between operating modalities.

In addition, there are a number of pan-system standards which should also be followed when building user centric software. These guidelines and standards have been created so as to encapsulate knowledge and best practice into a format which can be easily applied by engineers who do not have an in-depth knowledge of human factors or ergonomics. The standards have been designed by both operating system manufacturers and international standards bodies, and by following these standards the developer can be assured that their system will be usable by the most number of people.

Questions to think about as you design your prototype:

1. Am I developing a consistent interface?
2. Are the interactions consistent across the platform and development?
3. Is my command and event structure universal across the development and platform?
4. Am I following standards and best practice?
5. Am I following the platform design guide[32]?

Facilitate Familiarity

Familiarity (including via Metaphor) is a key principle for enhancing usability. As we have already seen from the Xerox Star team, and from many usability experts, dovetailing into a user's learned behaviour is the primary way of enhancing familiarity with a system, which they are as yet unfamiliar with. You can do this in many ways, from mapping real-world to virtual world concepts, using common terms and jargon, by making the look and feel of the new system similar to that of an old system – or an old manual way of doing things (such as forms which look similar). However you decide to implement familiarity, or enhance that familiarity within your system, always make sure you are thinking if your concept of familiarity are likely to be the same as your users. Remember the development is not for you, but for your target users.

Questions to think about as you design your prototype:

1. Does your system map real world concepts to the virtual world?

[32]Such as those for the Mac OSX http://developer.apple.com/library/mac/#documentation/UserExperience/Conceptual/AppleHIGuidelines/Intro/Intro.html

2. Does your system use terms the user is familiar with (including Jargon)?
3. Does the system work in familiar ways, with reference to itself and other comparable applications?
4. Do you assuage 'intuition' for familiarity?
5. Does your system use easily understandable (and therefore familiar) messages?

Interaction Stability

You won't find the principle of interaction stability in many other texts, but it seems to me it is a reasonable summary of some of the aggregated principles and guidelines which other usability experts purport. When we are thinking of interaction stability we are trying to make sure that interactions are stable, that they can be resumed if they break, that data will not be lost, that the place someone is at, in an interaction (the stage for the step), can be returned to in the event of a failure. For example, a web form which times-out after one hour and does not save the existing contents of that form does not exhibit interaction stability; however a form which times-out, saves the data, and allows the user to commence from this saved point, does exhibit interaction stability.

One further way of enhancing interaction stability is preview. A lack of preview of upcoming information is one of the major issues to be addressed when building interfaces and considering interaction stability. Consequently, preview is considered to be a primary component of good interaction stability. This preview can be manually achieved by try operations and awaiting desired or undesired outcomes from the environment. In an interface context, the lack of previews of both the outcomes of specific actions and information relating to navigation of the interface itself suggests that some degree of 'foreknowledge' should be implemented so that a limited preview can be obtained. In this case, a good interface design will remove the onus on the user by providing outcomes and descriptions of what will occur if a specific action is performed, as well as suggesting paths to certain information of functional outcomes.

Questions to think about as you design your prototype:

1. Are you able to resume interrupted actions?
2. Are you able easily reverse an action, incorrectly taken?
3. Are you able to understand your location in the interaction?
4. Does your system recover well from an unexpected event?
5. Does your interactions (including dialogs) exhibit stable non-cyclical behaviour and closure?

Facilitate Learnability

Learnability is a key aspect for interface design. Indeed, this is why large computer systems, and operating environments often come with a series of rules and best practices for software development which describe the standard ways of interacting on that specific system. Application menus which normally have 'file', and 'edit', to the left and 'help' to the far right all the way through to common

dialogues for picking colours or interacting with files systems, all try to enhance learnability. I have not yet met a user who has read the user manual for any large system. In reality, users prefer to investigate and orientate themselves to the system. Therefore, interface development should assist the user in self-directed learning of the interface components as opposed to relying on the reading of the user manual or help text. It is only by helping make your system easily learnable that you will be able to reduce the amount of support required by the user and therefore reduce long-term costs while enhancing user satisfaction.

Questions to think about as you design your prototype:

1. Is your system behaviour predictable?
2. Can users easily transit from novice to expert?
3. Can you understand the system behaviour without recourse to manuals of help systems?
4. How easy is it to learn any bespoke system functionality?
5. Does your system facilitate self learning, and functionality investigation?

Facilitate Robustness

As with all aspects of a computer system, speed of service balanced with robustness of the system, is critical for the user. In most cases, users prefer systems to be more robust because the effect of a faster service speed balanced against a system which crashes frequently. This scenario costs more time in the long run and introduces a lack of trust in the user's perception of the interface and therefore the quality of the end product. Obviously, robustness is a systemwide concern however, because the interface elements are often developed as one of the last things in the development lifecycle, robustness is sometimes sacrificed for a speedy development which ends on time.

Questions to think about as you design your prototype:

1. Does your system recover well from an unexpected event?
2. Are errors easily dealt with?
3. Are incorrect user actions easily recoverable?
4. Is the user-state saved between sessions in the event of a system failure?
5. How does your system handle abnormal input?

Facilitate Progressive Disclosure

Progressive disclosure is a tricky concept to discuss. It was perviously thought that usability was enhanced when the number of selections required of a user was reduced to the bare minimum. This created interfaces that were very complex having as much functionality as possible on the screen at the same time. In reality, quantitative measures showed that this decreased performance because it took the user longer to make a decision about what to select as opposed to the selections themselves. Progressive disclosure is an antidote to this kind of information and operation overload and suggests

that a hierarchy of operations moving through simple selectable steps is faster and easier for the user to understand than counting the selections required to achieve a specific action. Of course progressive disclosure can lead to problems because there may be multiple levels of disclosure, in which case the functionality can be hidden in the depths of the computer system such that it is never used. The interface developer therefore needs to understand these complexities and try to mitigate them by providing progressive disclosure but limiting the hierarchical nesting of the activities that are being disclosed to a minimum.

Questions to think about as you design your prototype:

1. Does your interface look overly complex? If so, simplify.
2. Are there a lot of components displayed at one time? If so clean it.
3. Are there a multitude of possible actions available to the user? If so focus on building one action for one interface element.
4. Is there a tight logical hierarchy of actions?
5. Is the user led along the interactive path?

Facilitate Scalability

Again, scalability is not a principle you will find in most usability texts, however it is important in the real world. If your system is not able to scale with regard to the way it displays data, or the way interactions progress, in a system which is being progressively overloaded, then the efficiency of that system, its usability, is drastically reduced. For example, I once worked on a system which displayed data in interactive bubbles, this worked well with small amounts of data – and also looked supercool. However, as the amount of data was increased the interface was not scalable and so it became unusable, the amount of interactive bubbles expanding exponentially – making visual search very difficult. The solution was to define the point whereby the display became unusable, and then adapt the data back into a more traditional tabular format which was far easier to search and could efficiently display much more data than the interactive bubble format – although it didn't look as cool.

Questions to think about as you design your prototype:

1. Does your interface scale to handle larger datasets then envisaged etc?
2. Does your system handle data and interaction within an acceptable time?
3. Do complex actions scale up in terms of data and user requirements?
4. Do your interfaces remain simple when information is being dynamically added?
5. Can new functionality be added to your system without negatively impacting on its current interactions and interfaces?

Facilitate Self-Description

The principle of self-description is one which is pointed to by some usability experts, and is implicit in other usability principles; such as those which espouse good help and good documentation. The difference with self-description is that in the best possible circumstance the user should not need to refer to help or the documentation. This is because it is very unlikely – indeed normally only in extremes – that a user will ever consult the manual. One thing you should consider is that self-description is not necessarily only about explicit textual descriptions. Implicit descriptions based on the visual rendering – the way something looks – dovetailing into familiarity and simplicity, are also important in the self-description principle. By visually suggesting a path the user should take you enhance the self-description of that particular interface and interaction.

Questions to think about as you design your prototype:

1. Is your system well documented?
2. Is help present and informative?
3. Is it possible to understand the program functionality without recourse to the manual?
4. Is it possible to understand the interface, widgets, and interactivity without recourse to the manual?
5. Is it possible to fully understand all dialogs, messages, and status'?

Facilitate Simplicity

Simplicity is a commonly espoused usability principle, indeed many usability experts list this as a key principle to be followed. However, from work that my team and I have conducted, building in simplicity treads a very fine line, because it impacts on aesthetics, pleasure, and emotion. An interface which is too simplistic will be seen as boring, or only utilitarian, one that is too complex will be perceived as being overly complicated and aesthetically displeasing. Simplicity of the interaction on the other hand is a useful trait to develop, and this can be done by hiding complex system functionality from the user (often the novice user), thereby presenting them with a constrained set of choices, while still enabling the complexity to be accessed by expert users.

Questions to think about as you design your prototype:

1. Is your system presented simply?
2. Are the interactive elements simple to understand and use?
3. Can you understand the system behaviour without recourse to manuals of help systems?
4. Does your system exploit natural interactive constrains?
5. Is complexity hidden from the novice user?

Facilitate Situational Awareness

Again, situational awareness is not a principle you will commonly come across in many usability texts, however I think it is a good way of describing certain aspects of a usable design. Let us recap: Situational awareness involves a perception of the environment critical to making decisions in complex and dynamic situations. Indeed, situation awareness *"involves being aware of what is happening in the vicinity to understand how information, events, and one's own actions will impact goals and objectives, both immediately and in the near future"*. Can increase our awareness of the situation and build this model then aspects which are common within the usability literature such as the desire for simple errors, informative feedback, visible components and interactions, coupled with a singularity of focus, all go towards creating an environment which enhances situational awareness, but situational awareness is actually much more than this; let us take a brief look at a model of situational awareness (see Figure: Endsley's Model of Situation Awareness).

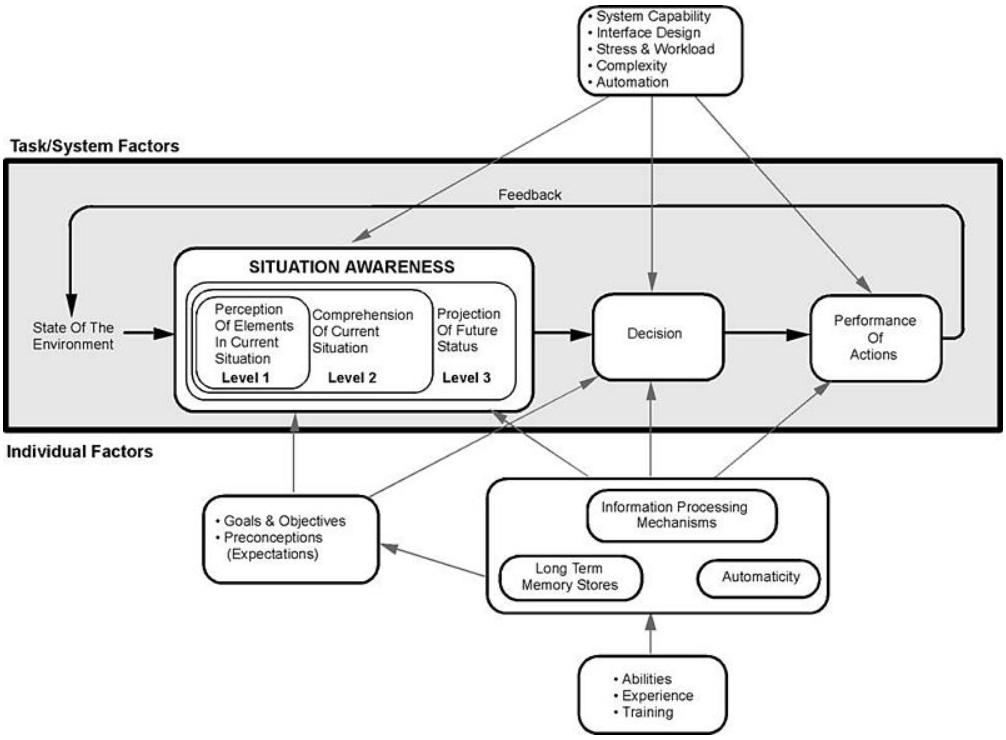

Figure: Endsley's Model of Situation Awareness. Endsley's Model of Situation Awareness —Image Credit: adapted from Endsley, 1995.

Endsley's Model espouses a good perception of the elements in the current situation, a comprehension of that situation, and an understanding of the future status. All of these factors: enable you to make a decision as to the interaction; perform these interactions which then alter the state of the environment – in this case the interface or the interaction; which then affects the perception of elements… so on. We can see that this kind of view is directly related to good usability. Understanding the interface and its abilities are all part of the perception of interface elements, a comprehension of the current interactive situation including your location in the interaction.

This is known as the user orientation or 'where-ness' (detecting cyclic behaviour, direction and distance) and is important in interface design as it enables users to navigate with some degree of accuracy. The environment however, must be updated such that cues are provided in an appropriate manner, giving explicit orientation information such that navigational information can be detected. The similarities between real-world movement and that surrounding the interface suggests that the provision of some form of explicit and appropriate orientation method are an advantage when navigating the interface. This would mean that a user can make a choice as to whether they want to be at the current location and if not aid them in deciding how to best navigate to their perceived destination. Finally, situational awareness rests upon our ability to predict our future status (this may be including familiarity as well as an understanding of how the system works). As an example, think of the last time you completed a credit card transaction or an order from the likes of 'Amazon'. In this case, you are led through a sequence of events; each step is given in graphical overview at the top of the page, and as each step is accomplished so the colour is changed to highlight a stage has been satisfactorily completed (see Figure: Amazon Check Out). This is all about enhancing your situation awareness by providing you with a stepwise overview of the current situation and an understanding of what will happen in the future.

Figure: Amazon Check Out. Amazon Check Out – 'Projection of Future Status' —Image Credit: Amazon.

Questions to think about as you design your prototype:

1. Does your system facilitate orientation both within the interface and within the interaction?
2. Is orientation and navigation, around and through the interface (and interaction), easy?
3. Is error handling simple? Is feedback informative?
4. Are all components, needed for the interaction, visible?
5. Do you maintain a single focus of interactive attention, without distractors?

By understanding and following these simple design principles any software developer can begin to create user centric applications which will be implicitly friendly to users. By applying these principles through requirements elicitation, analysis, user modelling, and interface development, user facing components can be enhanced and best practice can be implicitly included within the development.

Remember… these are just the next ten of our total principles - for the next batch you can skip ahead). But just because you can, maybe you shouldn't… take your time to read the rest!

Summary

We can see that designing and building usable and efficient interfaces is by no means a trivial task. Even with the wealth of toolkits and accessibility API's currently available the software engineer and

UX specialist should not underestimate the amount of work and testing required for a comprehensive interactive development. Fortunately there are a number principles to assist in this process, as well as the basic elements of the graphical user interface which can be assembled into a form which directly supports usability.

Some researchers see the ubiquity of the graphical user interface as being detrimental to the HCI field in general. They claim that focusing only on this kind of interaction means that there has been no novel work undertaken in the HCI domain since the heady days of Parc Xerox. I have some sympathy for this view but in reality it is not completely correct. With the development of gesture-based interfaces and the upcoming use of haptics the interactive elements of interface design are alive and well. I would agree, however, that the interactive displays used in these systems are stagnating to some degree. But in the context of development, building and designing interactive components which are tailored to the users need is difficult enough. By leveraging the principles already undertaken and the toolkits which provide well understood and consistent components and metaphors the software engineer can be reasonably assured that the interface design will be very familiar to the user and in this way assist usability in a practical manner.

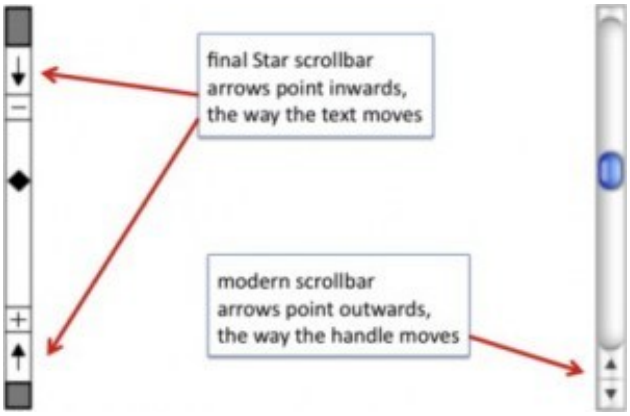

Figure: Scrollbar Miss-Design. Xerox Star and Modern Mac OS X Scrollbars —Image Credit: Byte.

Before we finish, a word of warning – as always from Dix – who suggests we may often ignore our fields previous work and start to reinvent the wheel:

> *"When one builds the justification of why something should work, the argument will not be watertight in the way that a mathematical argument can be. The data on which we build our justification has been obtained under particular circumstances that may be different from our own, we may be bringing things together in new ways and making uncertain extrapolations or deductions. Some parts of our argument may be strong and we would be very surprised if actual use showed otherwise, but some parts of the argument may involve more uncertain data, a greater degree of extrapolation or even pure guesswork.*
>
> *These weaker parts of the argument are the ideal candidates for focusing our efforts in evaluation. Why waste effort on the things we know anyway; instead use those precious*

> *empirical resources (our own time and that of our participants) to examine the things we understand least well. This was precisely the approach taken by the designers of the Xerox Star. There were many design decisions, too many to test individually, let alone in combinations. Only when aspects of the design were problematic, or unclear, did they perform targeted user studies. One example of this was the direction of scroll buttons: should pressing the 'up' button make the text go up (moving the page), or the text go down (moving the view)?*
>
> *If there were only one interpretation it would not be a problem, but because there was not a clear justification this was one of the places where the Star team did empirical evaluation… it is a pity that the wrong answer was used in subsequent Apple Lisa design and carried forward to this day."* (see Figure: Scrollbar Miss-Design).

Indeed, in conversations with David Smith at CHI98 Dix described how in the first version of the design documents for the Star

> *"the scrollbar arrows pointed outwards as they do in modern interfaces. However, unsure of the correct orientation, the Star design team performed user studies with both orientations. Whereas the software designers were quite happy with the outwards form, the non-computing users were uniformly confused by this direction of arrows, hence the inwards pointing arrows were adopted for the final Star design.*
>
> *Unfortunately when the Star design documents were passed on to the later design teams for the Lisa and Macintosh, the initial, wrong version of the scrollbar designs was used! Hence we came by our current scrollbar arrow direction by accident and it is precisely the opposite of what was found to be easy to use"* [33].

Optional Further Reading

- [S. K. Card,] T. P. Moran, and A. Newell. The psychology of human-computer interaction. L. Erlbaum Associates, Hillsdale, N.J., 1983.
- [J. Johnson.] GUI bloopers 2.0: common user interface design don'ts and dos. Elsevier/Morgan Kaufmann Publishers, Amsterdam, updated and rev., edition, 2008.
- [J. Nielsen.] Usability engineering. Academic Press, Boston, 1993.
- [H. Sharp,] Y. Rogers, and J. Preece. Interaction design: beyond human-computer interaction. Wiley, Chichester, 2nd ed edition, 2007.
- [B. Shneiderman] and C. Plaisant. Designing the user interface: strategies for effective human-computer interaction. Addison-Wesley, Boston, 5th ed edition, 2010.

[33] http://www.guidebookgallery.org/articles/handsacrossthescreen.

International Standards

- [ISO/TR 9241-100:2011.] Ergonomics of human-system interaction – part 100: Introduction to standards related to software ergonomics. TC/SC: TC 159/SC 4 ICS 13.180; 35.180, International Organisation for Standardisation (ISO), Geneva, Switzerland, 2011.
- [ISO/TR 9241-110:2006.] Ergonomics of human-system interaction – part 110: Dialogue principles. TC/SC: TC 159/SC 4 ICS 13.180, International Organisation for Standardisation (ISO), Geneva, Switzerland, 2006.

Self Assessment Questions

Try these without reference to the text:

1. What is the significance of the Xerox Star interface?
2. What are the five main principles proposed by the Xerox Star team?
3. What does GOMS stand for and what does it involve?
4. What are the ten main principles of efficient design?
5. How do these principles differ from Shneiderman's rules?